Wagner, Philipp

Grafikbasierte Objekterkennung mit HP QuickTest Professional

eingereicht als

BACHELORARBEIT

an der

HOCHSCHULE MITTWEIDA (FH)

UNIVERSITY OF APPLIED SCIENCES

Fachbereich Informationstechnik und Elektrotechnik

Mittweida, 2009

Erstprüfer: Prof. Dr.-Ing. Frank Zimmer
Zweitprüfer: Michael Schröer

Bibliographische Beschreibung

Wagner, Philipp:
Grafikbasierte Objekterkennung mit HP QuickTest Professional. 2009. - 78 S. Mittweida, Hochschule Mittweida (FH), Fachbereich Informationstechnik und Elektrotechnik, Bachelorarbeit, 2009

Referat:
Ziel dieser Bachelorarbeit ist es, die Probleme der Objekterkennung, die durch Hewlett Packard QuickTest Professional auftreten, zu erläutern und eine entwickelte alternative Funktionalität vorzustellen, die eine neue Herangehensweise zur Erkennung der Testobjekte bereitstellt. Diese Funktionserweiterung beruht auf der tatsächlichen grafischen Ausgabe am Bildschirm und stellt anhand eines Vergleichsmusters fest, ob und wo bestimmte Elemente in einer Anwendung zu sehen sind, unabhängig von den Aussagen der Hewlett Packard QuickTest Professional Objekterkennung.

Inhaltsverzeichnis

1. Einleitung

Heutzutage nehmen Computer mit zunehmendem Maße eine immer größere Bedeutung in unserem Leben ein. Eines der Schlüsselelemente für den erfolgreichen Betrieb von Computern ist die Software. Software-Systeme werden immer umfangreicher, komplexer, anspruchsvoller und sind mittlerweile ein wesentlicher Bestandteil im privaten, öffentlichen und wirtschaftlichen Bereich. Einsatzgebiete von Software-Systemen finden sich beispielsweise in der Medizin, in der industriellen Produktion, verbunden mit einem hohen Grad an Automatisierung, spielen in speziellen Bereichen, wie der Flugsicherung, dem Verkehrswesen oder dem Energiewesen eine sehr große Rolle, sind ein unverzichtbares Hilfsmittel in der Verwaltung bei Banken, im Versicherungswesen und im Handel, stellen die Basis für umfangreiche Kommunikationssysteme und erleichtern uns den Alltag in vielen Bereichen.
Aufgrund der zunehmend höheren Komplexität der Software-Systeme werden jedoch nicht geplante und unvorhersehbare Wechselwirkungen immer wahrscheinlicher. Dabei kann es bei fehlerhafter Software zu unterschiedlich schweren Schäden kommen, wie beispielsweise Geld-, Zeit- oder Imageverlust, in einigen Fällen sogar zu Personenschäden. Einige bekannte Software-Fehler sollen das verdeutlichen:

- Bei der ersten Demonstration des neuen Betriebssystems Windows 98 auf der Computermesse Comdex in Chicago erschien dem Microsoft-Chef Bill Gates eine jener Fehlermeldungen, die von Millionen seiner Kunden weltweit so gefürchtet sind. Danach stürzte der PC vor den laufenden Kameras ab. Ein peinlicher Auftritt für den Software-Milliardär und ein Imageschaden der nur schwer zu beheben ist. [1]
- Technische Probleme beim Mautsystems der Firma Toll Collect führten zu Verzögerungen bei der Einführung des Systems. Mit 16 Monaten Verspätung konnte das System Anfang 2005 in Betrieb genommen werden. Die Einnahmeausfälle betrugen ca. 3,5 Milliarden Euro. [2]
- Im Jahre 1983 übten Jagdbomber vom Typ F18 mit neuer Bordsoftware. Bei Testflügen, bei dem der Äquator überflogen wurde, drehten sie sich auf den Kopf. Die Ursache war ein Vorzeichenfehler in einem Programm. [3 S.1]
- 1984 gab es eine Überschwemmung im südfranzösischen Tarntal, weil ein Computer des automatischen Sperrwerks bei Réquista die Falschmeldung einer Überlaufgefahr nicht erkannte und zwei Schleusentore öffnete. [3 S.1]

Anhand dieser Fakten ist es sinnvoll, sich verstärkt mit der Sicherung der Software-Qualität zu beschäftigen. Dabei kann das Testen von Software-Systemen das Risiko, dass Probleme im operativen Betrieb auftreten, reduzieren und die Qualität der Software erhöhen, indem Fehler vor der Übergabe des Systems im Entwicklungsprozess identifiziert und behoben werden. Die Herausforderung besteht dabei, die Qualität der Software in einem angemessenen Umfang, in angemessener Zeit und zu angemessenen Kosten bei vertretbarem Risiko zu sichern.

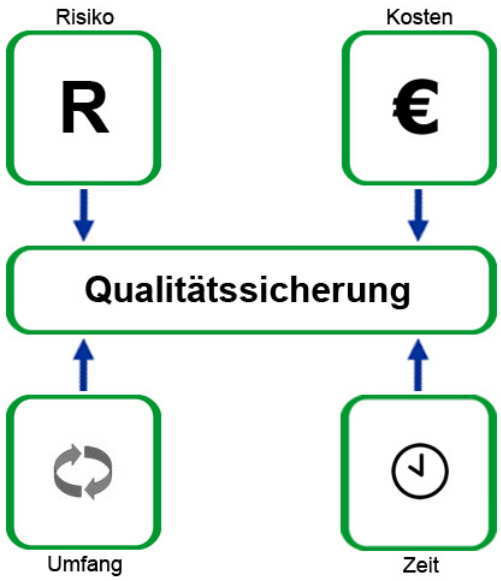

Abbildung 1 Qualitätssicherungsmodel

Eine gewinnbringende Möglichkeit, diesem Problem zu begegnen, ist die Automatisierung der einzelnen Testaktivitäten zur Qualitätssicherung. Die dafür angebotenen Functional-Testing-Tools, die in der Phase der Testdurchführung zum Einsatz kommen, automatisieren dabei Funktions- und Regressionstest und unterstützen damit den in der Regel umfangreichsten und aufwändigsten Teil im Testprozess. Hewlett-Packard (HP) ist im Bereich der Functional-Testing-Tools Markführer und stellt mit der Testsoftware HP QuickTest Professional (HP QTP) eine hervorragende Lösung bereit, mit der automatisierte Funktions- und Regressionstests für alle wichtigen Softwareapplikationen und Umgebungen durchgeführt werden können. Diese Lösung nutzt das Konzept des automatisierten Testens mithilfe von Schlüsselwörtern und erleichtert so das Erstellen und Verwalten von Tests, wobei die Testfälle mithilfe einer bestimmten Erfassungsmethode erstellt werden, bei der Abläufe und Objekte direkt aus den Applikationsbildschirmen erfasst werden. Doch gerade bei Anwendungen, die

asynchron zum Anwender im Hintergrund arbeiten (z.B. AJAX-Webanwendungen), kann es zu Problemen mit der Objekterkennung durch HP QTP kommen.

Das Ziel dieser Arbeit ist es, die Probleme der Objekterkennung, die durch HP QTP auftreten, zu analysieren und eine alternative Funktionalität zu entwickeln, die eine gänzlich andere Herangehensweise zur Erkennung der Testobjekte bereitstellt und auf der tatsächlichen grafischen Ausgabe am Bildschirm beruht. Anhand eines vorgegebenen Vergleichsmusters ist festzustellen, ob und wo ein bestimmtest Element in einer Anwendung tatsächlich zu sehen ist, unabhängig von den Aussagen der HP QTP Objekterkennung. Dazu wird im ersten Teil dieser Arbeit dem Leser ein Überblick über den Qualitätsprozess, deren Merkmale und Maßnahmen, sowie den enthaltenen Testmethoden und dem Testprozess vermittelt. Des Weiteren wird auf die Automatisierung der einzelnen Testaktivitäten eingegangen, wobei der Schwerpunkt dieser Arbeit auf die werkzeugunterstütze Testautomatisierung der Funktions- und Regressionstests gelenkt wird. Im weiteren Verlauf wird der Fokus speziell auf die Testsoftware HP QTP gelegt und deren Aufbau und Funktionsweise erläutert. Darüber hinaus wird eine Analyse des Testtools durchgeführt, bei der insbesondere die fehlerhafte Objekterkennung im Vordergrund steht. Im Anschluss der Analyse folgt die Präzision der Ergebnisse, die Ableitung eines möglichen Lösungsansatzes für die fehlerhafte Objekterkennung und die Lösungsstrategie. Danach wird im Kapitel „Vorstellung einer alternativen Objekterkennung“ die Realisierung der Lösungsstrategie detailliert beschreiben, wobei insbesondere der theoretische Ansatz, die Voraussetzungen zur Realisierung, die praktische Umsetzung und die Gesamtbewertung des Lösungsansatzes genauer betrachtet wird.

Anhand dieser Vorgehensweise soll die professionelle Automatisierungssoftware HP QTP erweitert, die automatisierten Funktions- und Regressionstests verbessert und damit die Qualität eines Software-Systems erhöht werden.

2. Der Qualitätsprozess

2.1. Qualitätsbegriff

Die Bezeichnung Qualität ist im allgemeinen Sprachgebrauch bekannt und bezieht sich auf die Beschaffenheit oder die Eigenschaft eines Produkts. Oftmals wird durch hinzufügen von Adjektiven die Wertung des Begriffs beeinflusst. Man spricht beispielsweise von schlechter, guter oder ausgezeichneter Qualität. Doch wann ist die Qualität eines Produkts gut und wann ist diese schlecht?

Als Grundlage dient die Norm EN ISO 9000:2005, die den Begriff Qualität folgendermaßen definiert: "Qualität ist der Grad, in dem ein Satz inhärenter Merkmale Anforderungen erfüllt." [4] Die Qualität gibt also an, in welchem Maße ein Produkt den bestehenden Anforderungen entspricht. Das Wort inhärent bedeutet einer Sache anhaftend oder einer Einheit innewohnend, insbesondere als ständiges Merkmal. Damit sind objektiv messbare Merkmale gemeint.

Diese Normserie beabsichtigt, dass die Qualität in Relation zu den Anforderungen bewertet und beurteilt wird, wobei eine große Zahl von Merkmalen und Eigenschaften zur Bestimmung der Qualität einfließen können. Wurde früher jedoch die Qualität nur als eine Eigenschaft von Produkten oder Dienstleistungen verstanden, also die Erfüllung der Kundenspezifischen Anforderungen, erstreckt sich der Qualitätsbegriff heutzutage, als umfassende Variante des Qualitätsmanagements, über das gesamte Unternehmen. Zur Erfüllung der Qualität treten also neben den Kundenanforderungen auch die Anforderungen der Mitarbeiter, Kapitalgeber und der Öffentlichkeit in den Vordergrund.

2.2. Qualitätsanforderungen

Die größte Herausforderung besteht darin, die geforderten Qualitätsanforderungen eines Software-Produkts in einem optimierten Qualitätsprozess umzusetzen, damit einerseits die Kosten zur Entwicklung gesenkt werden können, andererseits der Zeitaufwand zur Herstellung minimiert wird und dabei das Risiko von Defekten in der Produktion vermindert, also die Qualität verbessert und die Performance des Produkts gesteigert werden kann. Durch eine optimale Gestaltung dieser drei Faktoren (Risiko, Kosten, Zeit) können Unternehmen ihren Wettbewerbsvorteil erhöhen. Nicht nur die benötigte

Zeit zur Entwicklung der Software kann minimiert werden, wodurch das Produkt schneller am Markt und speziell beim Kunden verfügbar ist. Die Kosten werden dabei gesenkt und der Umsatz wird durch die frühere Einsetzbarkeit des Software-Produkts gesteigert. Dabei ist aber immer zu beachten, dass die Qualitätsanforderungen in einem angemessen Umfang im Entwicklungsprozess untersucht werden, damit das Risiko von Defekten und Fehlern reduziert werden kann.

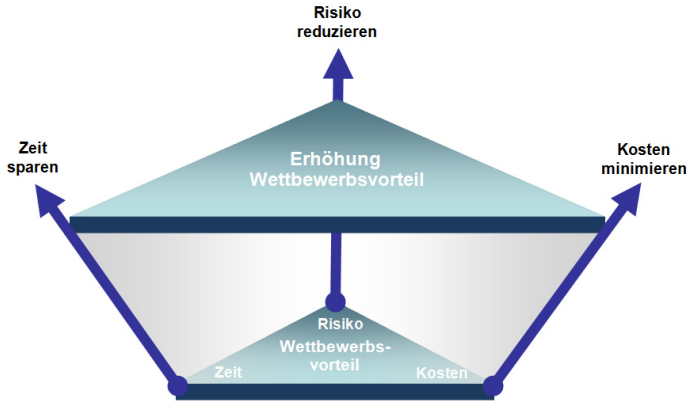

Abbildung 2 Optimierung der Qualitätsanforderungen [5]

Die steigende Komplexität und der hohe Integrationsgrad moderner Software-Systeme, die komplexen Schnittstellen zwischen den Komponenten, die Forderungen nach ausreichender Benutzerfreundlichkeit und Zuverlässigkeit insbesondere bei interaktiven Systemen erhöhen jedoch die Entwicklungskosten, führen aufgrund des Leistungsumfangs oft zum Terminverzug und erhöhen das Risiko der Fehlerentstehung im Entwicklungsprozess.

Von den Entwicklern wird also einerseits die strickte Einhaltung der Kosten-Zeitpläne für die Entwicklungsprojekte verlangt und andererseits eine hohe Qualität der Software-Produkte gefordert. Doch welche Qualitätsmerkmale muss ein Software-Produkt erfüllen? [5]

2.3. Qualitätsmerkmale

Zur Sicherung der Software-Qualität stellt die ISO/IEC 9126 Norm ein Model zur Verfügung, das sich auf die Produktqualität bezieht und die folgenden sechs Qualitätsmerkmale enthält:

- Funktionalität,
- Zuverlässigkeit,

- Benutzbarkeit,
- Effizienz,
- Wartbarkeit und
- Übertragbarkeit.

Mit dem Qualitätsmerkmal Funktionalität wird überprüft, ob die Software die festgelegten Eigenschaften besitzt, bzw. die geforderten Funktionen und definierten Anforderungen erfüllt. Des Weiteren wird die Sicherheit, Ordnungsmäßigkeit, Angemessenheit und Richtigkeit der Software sichergestellt. Die Zuverlässigkeit überprüft, wie es der Name schon sagt, die Fähigkeit der Software ein bestimmtes Leistungsniveau unter festgelegten Bedingungen über einen festgelegten Zeitraum zu bewahren. Dabei wird speziell die Verfügbarkeit, die Robustheit, die Wiederherstellbarkeit, die Stabilität und die Reaktionssicherheit in Fehlerfall untersucht. Der erforderliche Aufwand zur Benutzung der Software, der durch eine individuelle Beurteilung einer festgelegten oder vorausgesetzten Benutzergruppe beurteilt wird, ist Bestandteil des Qualitätsmerkmals Benutzbarkeit. Dabei werden die Verständlichkeit, die Erlernbarkeit, die Bedienerfreundlichkeit und die Dokumentation, sowie die Transparenz des Produktes genauer durchleuchtet. Ein weiteres Qualitätsmerkmal ist die Effizienz, bei der das Verhältnis zwischen dem Leistungsniveau der Software und dem Umfang der eingesetzten Betriebsmittel unter festgelegten Bedingungen betrachtet wird, wobei das Zeit- und Verbrauchsverhalten, die Konformität und die Performance eine große Rolle spielen. Der Aufwand, der zur Durchführung vorgegebener Änderungen notwendig ist, fällt dabei unter der Begriff Wartbarkeit. Zu den Änderungen können Korrekturen, Verbesserungen oder Anpassungen an der Umgebung, der Anforderungen oder der funktionalen Spezifikationen gehören, wobei der Fokus auf die Wartungsfreundlichkeit, Pflegbarkeit, Modifizierbarkeit und Testbarkeit gelegt wird. Mit dem letzten Qualitätsmerkmal wird die Eignung der Software, die von einer Umgebung in eine Andere übertragen werden soll, überprüft. Dabei kann es sich bei der Übertragbarkeit um eine organisatorische oder Hardware- bzw. Software-Umgebung handeln.

Zur Erfüllung dieser Merkmal und der Sicherung der Qualität eines Software-Produkts stehen eine Reihe von Qualitätsmaßnahmen zur Verfügung, die im nächsten Abschnitt genauer betrachtet werden. [6, 7]

2.4. Qualitätssicherungsmaßnahmen

Für die Sicherung der Software-Qualität stehen eine Vielzahl von Testmethoden zur Verfügung, die Anhand ihrer Prüftechnik unterschieden werden können.

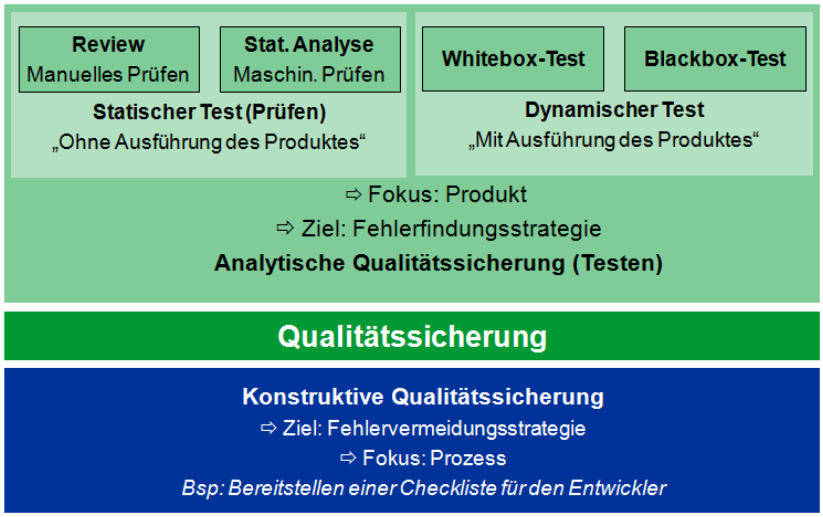

Abbildung 3 Übersicht – Maßnahmen zur Qualitätssicherung [17]

Diese Qualitätssicherungsmaßnahmen können dabei in den verschiedenen Phasen der Software-Entwicklung eingesetzt werden, wobei die „konstruktive Qualitätssicherung“ vorrangig in den frühen Projektphasen eingesetzt wird und den Schwerpunkt auf den Prozess der Entwicklung legt. Ziel ist es, von vorn herein Fehler zu vermeiden, im Gegensatz zur „analytischen Qualitätssicherung“, bei der die Fehlerfindung im Vordergrund steht. „Analytische Qualitätssicherungsmaßnahmen“ fokussieren und testen das Produkt, wobei je nach Methode die Software zur Ausführung gebracht wird oder ohne Ausführung der Software unter dem Begriff Prüfen bewertet und analysiert wird. [9, 14, 17]

Konstruktive Qualitätssicherung

Die konstruktive Qualitätssicherung unterstützt Software-Entwicklungsprozesse mit einer Vielzahl unterschiedlicher Methoden und Techniken zur:

- Qualitativen Verbesserung der Produkte,
- Frühzeitigen Fehlervermeidung und -behebung,
- Umsetzung einer benutzerfreundlichen und anwenderorientierten Anwendung,
- Durchführung einer optimierten Produktentwicklung und -herstellung.

Ziel der konstruktiven Qualitätssicherung ist es, von vornherein für ein gewisses Maß an Qualität eines Produkts durch Vorgänge von Konstruktionstechniken und Richtlinien zu sorgen. Diese Strukturierung wertet die Planungsaktivitäten in frühen Projektphasen auf und kann die Qualität, sowie die Produktivität bis zu 50% steigern. [3, 8, 9]

Analytische Qualitätssicherung

Durch analytische Qualitätssicherungsmaßnahmen wird die Qualität von Software geprüft und bewertet. Diese Maßnahmen können gegebenenfalls zu einer Verbesserung des Produkts führen. Es gibt eine Vielzahl von Gründen, warum diese Prüfungen durchgeführt werden sollten:

- Der Auftraggeber möchte wissen, ob ein Softwareprodukt seine Anforderungen erfüllt.
- Der Projektleiter ist an der Qualität der Zwischen- und Endprodukte einer Phase interessiert.
- Der Entwickler möchte wissen, ob er qualitativ gute Arbeit geleistet hat.

Eine wesentliche Voraussetzung für die analytische Qualitätssicherung ist eine funktionierende Qualitätsplanung. Diese legt fest, welche Anforderungen an das Produkt gestellt werden. Daraus lassen sich später die Prüfziele ableiten.
Es reicht nicht aus, erst nach der Entwicklung eines Softwaresystems festzustellen, ob diese die geforderten Merkmale besitzt und die spezifischen Anforderungen erfüllt. Die Prüfung der Qualität der Software ist nicht nur eine Prüfung von Zwischen- und Endprodukten, sondern auch eine Bewertung des Entwicklungsprozesses. Es müssen die verschiedenen Repräsentationen des zu erstellenden Softwareprodukts von der Anforderungsdefinition bis zum einsatzfähigen System geprüft werden.
Jede Prüfung hat ein bestimmtest Ziel, das durch die Anforderungen an die Software bestimmt wird. Diese Ziele sind vielschichtig. Die zur Verfügung stehenden Prüfmethoden lassen sich dabei in zwei Kategorien einteilen, in statische und dynamische Prüfungen. [3, 8, 9]

Statische Software-Testverfahren

Statische Software-Testverfahren gehören zu den analysierenden Verfahren. Sie zeichnen sich dadurch aus, dass die Software bei diesen Tests nicht ausgeführt wird, im Gegensatz zu dynamischen Software-Testverfahren. Der statische Test ist eine

Überprüfung, Vermessung und Darstellung eines Programms oder einer Komponente bzw. eines Dokuments, das eine formale Struktur besitzt. Zu unterscheiden sind:

- Die manuelle Prüfung in Form eines Reviews und
- Die automatisierte Prüfung in Form einer statischen Analyse.

Ziel eines statischen Tests ist die Ermittlung von Fehlern und Abweichungen von den vorhandenen Spezifikationen. Es werden vor allem Fehlerzustände aufgedeckt. Das Resultat eines statischen Tests ist die Verbesserung des Programmcodes (Effizienz, Robustheit) aber auch eine verbesserte Kommentierung des Codes (Wartbarkeit). [3, 10, 11]

Dynamische Software-Testverfahren

Während bei statischen Verfahren die zu testende Software nicht ausgeführt wird, setzen dynamische Verfahren die Ausführbarkeit der Software voraus. Zu unterscheiden sind:

- Das Black-Box-Testverfahren (funktionsorientiertes Verfahren) und
- Das White-Box-Testverfahren (strukturorientiertes Verfahren).

Grundprinzip der dynamischen Verfahren ist die Ausführung der zu testenden Software mit systematisch festgelegten Eingabedaten (Testfälle). Für jeden Testfall werden zu den Eingabedaten auch die erwarteten Ausgabedaten angegeben. Die vom Testlauf erzeugten Ausgabedaten werden danach mit den jeweils erwarteten Daten verglichen. Bei Abweichungen liegt ein Fehler vor. Ziel ist also das Aufdecken von Fehlerwirkungen und die Optimierung laufzeitkritischer Bereiche. [3, 8, 12]

Black-Box-Testverfahren

Die Bezeichnung Black-Box-Test steht für eine Methode des Softwaretests , bei der das Testobjekt (der Prüfling) als schwarzer Kasten angesehen wird. Die genaue Beschaffenheit des Programms wird also nicht betrachtet. Dies bedeutet, dass Tests ohne Kenntnisse über die innere Funktionsweise und Ablaufstruktur des zu testenden Systems entwickelt werden. Nur nach außen sichtbares Verhalten fließt in den Test ein. Wesentliche Voraussetzung für die Anwendung von Black-Box-Tests ist eine genaue Spezifikation (funktionelle Beschreibung) des Testobjekts. Vielfach wird in diesem Zusammenhang vom funktionellen oder entwurfsorientierten Testen gesprochen.

Ziel ist es, ausgehend von formalen und informellen Spezifikationen, Testfälle zu entwickeln, die sicherstellen, dass der geforderte Funktionsumfang eines Testsystems eingehalten wird. Es wird also die Übereinstimmung eines Testsystems mit seiner Spezifikation überprüft. Das zu testende System wird dabei als Ganzes betrachtet. Nur sein Außenverhalten wird bei der Bewertung der Testergebnisse herangezogen.
Ein erfolgreich durchgeführter Black-Box-Test ist aber keine Garantie für die Fehlerfreiheit eines Testobjekts. Die in frühen Phasen des Testsystem erstellte Spezifikationen könnte unter Umständen spätere Detail- und Implementations-entscheidungen nicht abdecken.
Da die Entwickler der Black-Box-Tests keine Kenntnisse über die innere Funktionsweise des zu testenden Systems besitzen dürfen, ist es vorteilhaft, ein separates Team zur Entwicklung der Tests einzusetzen. In vielen Unternehmen sind einzelne Testabteilungen für diese Aufgaben zuständig oder werden von anderen spezialisierten Firmen übernommen. [3, 13, 15]

White-Box-Testverfahren

Der Black-Box-Test wird eingesetzt, um Fehler gegenüber der Spezifikation aufzudecken. Um Fehler in bestimmten Komponenten oder sogar der fehlerauslösende Komponente selbst zu identifizieren, ist dieser Test ungeeignet. Dafür werden White-Box-Tests benötigt. Denkbar ist auch, dass zwei fehlerhafte Komponenten vorübergehend ein scheinbar korrektes Gesamtsystem erzeugen. Ein erfolgreich durchgeführter Black-Box-Test, der nur das Außenverhalten bewertet, würde dann ein fehlerfreies System garantieren, obwohl dies nicht der Fall ist. Dies kann durch White-Box-Tests leichter aufgedeckt werden.
Der White-Box-Test wird als strukturelle Testmethode bezeichnet und setzt voraus, dass die Struktur des Testobjekts bekannt ist. Ein Anwender der White-Box-Testmethode verfügt also über Kenntnisse der inneren Funktionsweise des zu testenden Systems. Im Gegensatz zum Black-Box-Test ist bei diesem Test also ein Blick in den Quellcode gestattet. [3, 13, 16]

Vergleich der beiden Testverfahren Black-Box und White-Box

Im Vergleich sind White-Box-Tests wesentlich einfacher in der Durchführung als Black-Box-Tests. Der geringere organisatorische Aufwand und die Möglichkeit des

Testens einzelner Teilkomponenten und deren interne Funktionsweise sind die klaren Vorteile des White-Box-Tests. Jedoch kann die Erfüllung der Spezifikation nicht überprüft werden. Dies ist ein wesentlicher Nachteil des White-Box-Tests. Black-Box-Tests sind zur Verifikation des Gesamtsystems besser geeignet.
Die übliche Vorgehensweise ist daher der Entwurf einer Black-Box-Testmethode, welche soweit als nötig durch die White-Box-Testmethode ergänzt wird. Dabei werden die gewünschten Vorteile von Black-Box-Tests und White-Box-Tests weitgehend miteinander verbunden und gleichzeitig die unerwünschten Nachteile eliminiert. [3, 15, 16]

2.5. Der fundamentale Testprozess

Testen umfasst alle statischen und dynamischen „analytischen Qualitäts-sicherungsmaßnahmen“. Ein Test muss ausreichend Informationen liefern, damit zuverlässige Entscheidungen zur Freigabe eines Produktes getroffen werden können. Auf Grund von limitierten Ressourcen, dem begrenzten Rahmen des verfügbaren Budgets und der vorgegebenen Zeit ist in der Regel kein vollständiger Test möglich. Demzufolge ist die Bestimmung des Testumfangs und Testaufwands notwendig, damit das bestmögliche Testergebnis geliefert werden kann. Dabei muss der Test auf die Prioritäten abgestimmt werden und die Konzentration auf risikoreiche Bereiche (Risikobewertung) oder fehleranfällige Bereiche (Komplexitätsbewertung) gesetzt werden.

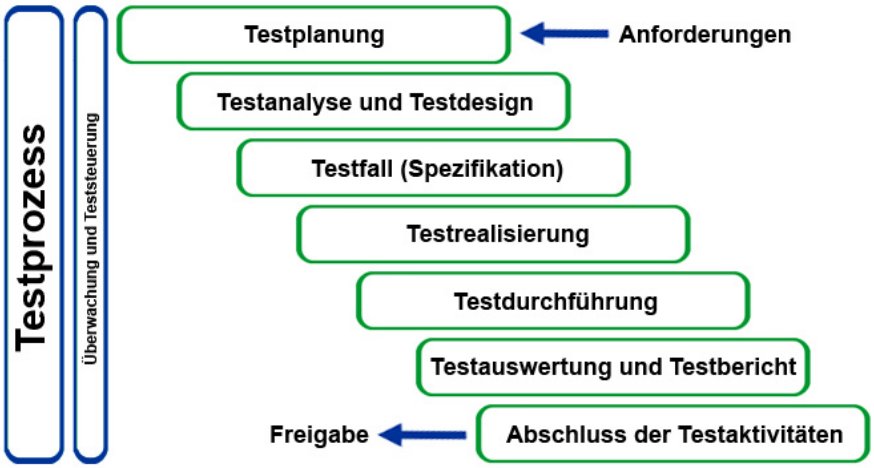

Abbildung 4 Der Testprozess und dessen Hauptaktivitäten

Der Testprozess ist fest im Software-Entwicklungsprozess eingebettet und unterteilt sich in sieben Hauptaktivitäten:

- Testplanung:

  Innerhalb dieser Aktivität wird ein Dokument (Testplan) erstellt, das die Testziele, den Testumfang, die Testverfahren bzw. Testmethoden, die Verantwortlichkeiten, die Ressourcen und Termine (Meilensteine) für alle beabsichtigten Testaufgaben enthält.

- Testanalyse und Testdesign:

  Testanalyse und Testdesign ist die Phase, in der die allgemeinen Testziele in konkrete Testbedingungen detailliert werden. Konkret muss erläutert werden, wie die im Testfall festgelegten Ziele zu erreichen sind, d.h. welche Methoden wie angewendet werden. Des Weiteren werden die benötigten Werkzeuge identifiziert und die Testumgebung organisiert.

- Testfall (Spezifikation):

  Auf der Basis der Testanalyse und dem Testdesign werden die Testfälle gebildet. Ein Testfall beschreibt in allgemeiner Form einen Testdatensatz oder eine Testsequenz und ist die Grundlage für die Spezifikation der Testdaten. Dazu gehören alle relevanten Angaben über die notwendigen Vorbedingungen, die Menge der Eingabewerte, die Menge der erwarteten Sollwerte und die erwarteten Nachbedingungen.

- Testrealisierung (Testvorgehensspezifikation):

  Die Testrealisierung dient zur Vorbereitung der Testdurchführung. Im Detail werden die Testfälle positioniert, die Testdaten erstellt, Testsuiten zusammengestellt, Skripte zur Testautomatisierung entwickelt, das Testsystem kontrolliert und die Konfiguration überprüft.

- Testdurchführung:

  Ausführung der Testfälle. Die Testergebnisse werden protokolliert und ein Vergleich Ist-Ergebnisse/Ist-Verhalten mit den erwarteten Soll-Ergebnissen/Soll-Verhalten ist durchzuführen. Gefundene Fehlerwirkungen oder Abweichungen werden festgehalten und analysiert. Ausgeführte Testfälle werden zu Testaktivitäten mit eingesetzten Testwerkzeugen, Testmitteln und getesteter Software-Version referenziert.

- Testauswertung und Testbericht:

Die Testergebnisse werden analysiert und zusammengefasst. Ziel ist die Bewertung des durchgeführten Tests und ein Entscheid, ob das Testende erreicht wurde, der Test zu wiederholen ist, oder der Test abgerochen werden muss.

- Abschluss der Testaktivitäten:
  Der Abschluss der Testaktivitäten umfasst die Kontrolle der Arbeitsergebnisse und die Erstellung von Statistiken zu Aufwand, Kosten und der Fehlerfindungsrate. Des Weiteren werden die Testmittel, die Testumgebung und die Infrastruktur für spätere Wiederverwendung dokumentiert, archiviert und gegebenenfalls der Wartungsorganisation übergeben. Abschließend findet die Testprozessanalyse statt um ein mögliches Verbesserungspotential für kommende Projekte zu dokumentieren.

Der gesamte Testprozess wird über die so genannte Teststeuerung ständig überwacht. Dabei werden der Testfortschritt analysiert, der Testfortschritt dokumentiert und falls notwendig Korrekturmaßnahmen eingeleitet. [3, 17]

2.6. Testautomatisierung

Manuell durchgeführte Test, die eine Vielzahl von Defekten in einer Softwareapplikation aufzeigen können, sind größtenteils sehr mühsam und zeitaufwendig. Testfälle und deren Testergebnisse werden manuell protokolliert und ausgewertet, wobei oftmals Fehlerdetails in der Masse der Testfälle übersehen werden. Des weiteren gleicht eine Testwiederholung oft nicht exakt dem vorhergehenden Test und muss mit einen etwa gleichbleibenden Aufwand durchgeführt werden. Die Testqualität besitzt dem entsprechend eine geringe Aussagekraft.

Die Testautomatisierung ist eine effiziente und kostensparende Möglichkeit die Qualität eines Softwaretests zu erhöhen. Testfälle können schneller entwickelt werden, sind gut dokumentiert und können unbeaufsichtigt ohne Benutzerinteraktion ausgeführt werden. Testdurchführungen werden beschleunigt durchgeführt, wodurch sich kürzere Testzyklen ergeben und können ohne Mehraufwand beliebig oft wiederholt werden. Durch exakt gleiche Testläufe (Wiederholungen) sind die Testergebnisse reproduzierbar und genauer. Darüber hinaus wird die Analyse der Testergebnisse durch eine automatisierte Reportgenerierung verbessert.

Zur Unterstützung der Testautomatisierung stehen viele Testwerkzeuge zur Verfügung. Diese sind sehr unterschiedlich und beziehen sich auf die verschiedenen Phasen des Software-Entwicklungsprozesses, wie beispielsweise der Testplanung und Verwaltung,

der Testanalyse und dem Testdesign, zum Aufbau der Testfälle (Spezifikation), der Testrealisierung, sowie der Testdurchführung und Beurteilung. Testwerkzeuge sollten jedoch mit Bedacht ausgewählt werden. Besonders wenn Testwerkzeuge in Betracht gezogen werden, die einen erheblichen finanziellen Beschaffungsaufwand bedeuten oder der effiziente Einsatz dieser Werkzeuge mit großem Mehraufwand verbunden ist. Deshalb müssen die Kriterien zur Auswahl der Werkzeuge sorgfältig auf die Testanforderungen abgestimmt werden.

Testwerkzeuge die in der Phase der Durchführung zum Einsatz kommen, werden als Functional-Testing-Tools, also als funktionsorientierte oder funktionale Testwerkzeuge bezeichnet. Dabei werden die Tests ohne die genaue Beschaffenheit des Programms entwickelt, sodass nur das nach außen sichtbare Verhalten und die Funktionalität überprüft wird. Funktional-Testing-Tools unterstützen also die Automatisierung der Black-Box-Testverfahren. Ziel dieser Werkzeuge ist es, Aktionen an der Benutzerschnittstelle aufzuzeichnen, so dass diese später wiedergegeben werden können. Zu diesem Zweck wird jede Interaktion vom Benutzer und der Softwareapplikation in Form eines Testskripts, das aus Aktionen und Ergebnisprüfungen besteht, aufgezeichnet. Die entstandenen Testfälle werden als Funktions- und Regressionstests bezeichnet. Während der Wiedergabe führen diese Werkzeuge die aufgezeichneten Testfälle durch, wobei die benutzerdefinierten Ereignisse und Aktionen mit dem tatsächlichen Verhalten der Applikation verglichen werden.

Testskripte zur automatischen Testdurchführung sind aber nur profitabel, wenn ein Test mehr als nur einmal durchgeführt wird. Ist dies der Fall entfalteten Funktions- und Regressionstests ihr gesamtes Potential. Automatische Test, die während der Software-Entwicklung eingesetzt und in definierten Zeitabständen durchgeführt werden, zeigen mögliche Nebeneffekte oder Folgefehler von vorgenommenen Änderungen direkt und erkennbar an. Sie dienen also als direkte Rückkopplung für den Entwickler, der unter Umständen nicht in der Lage ist, das Gesamtsoftwaresystem auf einmal zu überschauen. Doch nicht nur während der Entwicklung der Software ist der Einsatz von Funktions- und Regressionstests sinnvoll. Selbst nach der Fertigstellung und der Übergabe des Projekts kommt es immer wieder zu Änderung am fertigen Softwareprodukt. Modifikationen zur Anpassung und Weiterentwicklung, zur Korrektur oder Pflege der Software entstehen regelmäßig. Demzufolge unterstützen Funktions- und Regressionstests, bei Wiederholung in definierten Zeitabständen, die Fehlererkennung

und sichern die Korrektheit des Systems, wodurch sich die Qualität der Software verbessert. [5, 17, 18, 19, 20]

3. Werkzeugunterstützte Testautomatisierung mit HP QuickTest Professional

3.1. Einbettung von HP QuickTest Professional im Qualitätsmanagement-Prozess

Bei der Optimierung der Geschäftsprozesse unterstützt HP die Unternehmen mit einem umfassenden Business Technology Optimization (BTO) Softwareportfolio. Dabei ist HP mit einen Markanteil von mehr als 58% der weltweit führende Anbieter von BTO-Software. [21 S.33, 24, 25, 26] Das gesamte BTO-Softwareportfolio der Firma HP ist sehr umfangreich, komplex und bietet integrierte Softwarelösungen für die drei Gebiete Strategie, Applikationen und Betrieb. Im Bereich Applikationen stehen den Unternehmen professionelle Werkzeuge zur Verfügung mit denen die Qualität, Performance und Verfügbarkeit von Applikationen erhöht und die Problemlösungen verbessert werden. Gleichzeitig können alle IT-Projekte, die Risiken und Kosten sowie die Einhaltung der gesetzlichen Richtlinien überwacht werden. [25] Darüber hinaus ergänzen Technologien und Dienstleistungen zahlreicher internationaler Partner das BTO-Lösungsangebot von HP.

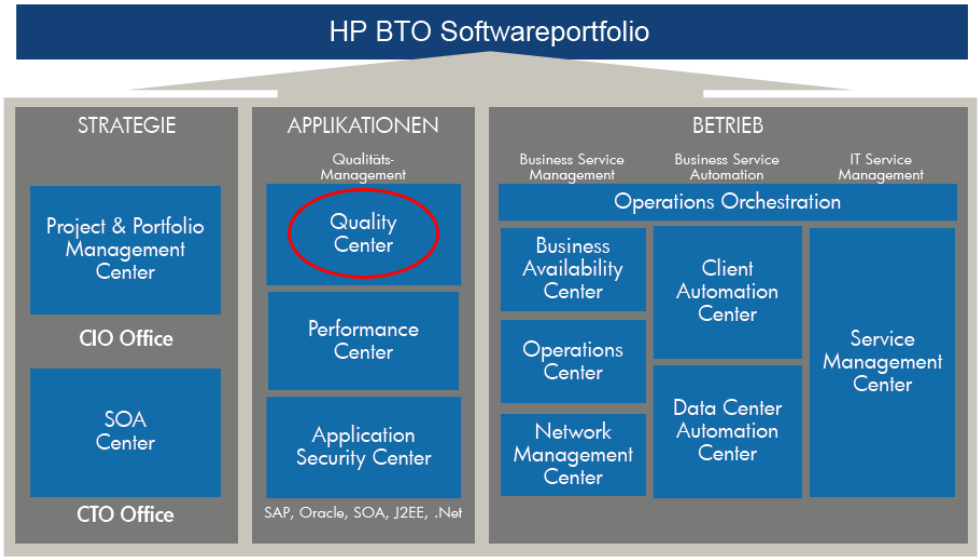

Abbildung 5 HP BTO Softwareportfolio mit dem Quality Center [21 S.4]

Mit einem der wichtigsten Bestandteile des BTO Softwareportfolios stellt HP mit dem HP Quality Center (HP QC) ein webbasierendes System zur automatisierten

Softwareprüfung und Qualitätssicherung bereit, mit der ein breites Spektrum an IT- und Applikationsumgebungen abdeckt werden kann. Durch Nutzung von konsistenten, wiederholbaren und standardisierten Verfahren werden Qualitätssicherungsprozesse wesentlich transparenter und die Qualität der Anwendungen kann während der Entwicklung optimiert und Risiken kontrolliert bzw. gesteuert werden. Darüber hinaus ermöglicht HP QC eine zentral koordinierte und überwachte Zusammenarbeit aller in die Software-Qualitätssicherung involvierten Teams und realisiert so neben deutlichen Zeit- und Kostenersparnissen auch wirksame Effizienzvorteile. Das HP QC vereint dabei integrierte Module zur Qualitätssicherung von Applikationen, wie das Functional-Testing-Tool HP QTP und ein Dashboard als zentrale, in Echtzeit arbeitende Überwachungs- und Steuerungskonsole. [28, 29]

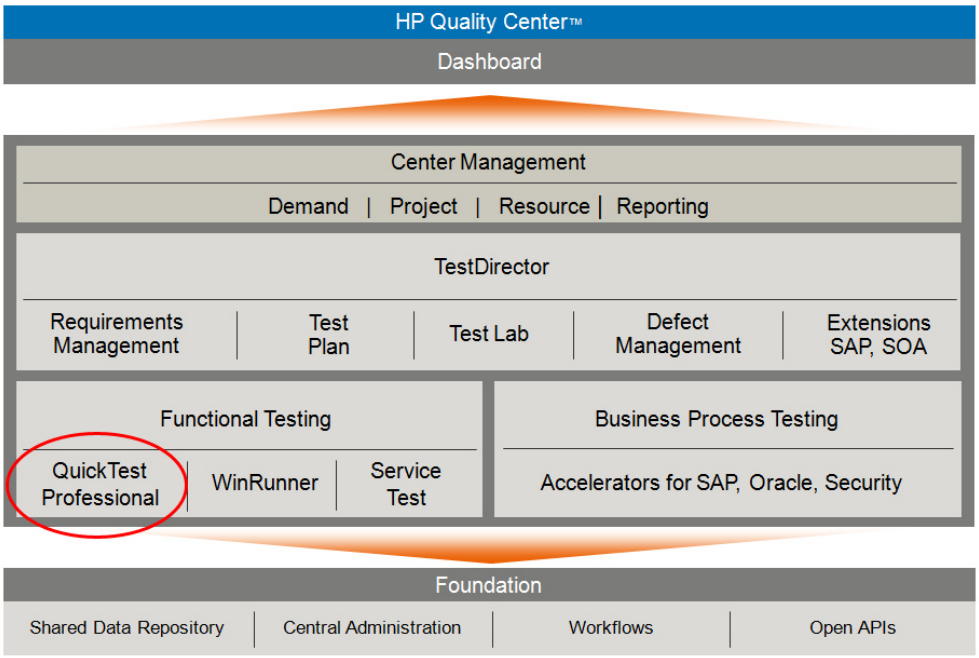

Abbildung 6 Übersicht HP Quality Center [21 S.19]

Dabei stellt HP QTP eine professionelle Lösung zur Automatisierung von Funktionstests zur Verfügung und unterstützt automatisierte Funktions- und Regressionstests für alle wichtigen Softwareapplikationen und Umgebungen. Das Erstellen von anspruchsvollen Testreihen, sowie die Dokumentation der Testfälle ist in einem Schritt realisierbar. Dabei werden die Testfälle mithilfe einer bestimmten Erfassungsmethode erstellt, bei der Abläufe direkt aus den Applikationsbildschirmen erfasst werden. Darüber hinaus wird eine schnellere Fehlerbehebung, mit Hilfe der

vollständigen Dokumentation und Replikation der auftretenden Fehler, für die Entwickler ermöglicht. [30]

Zum besseren Verständnis folgt im weiteren Verlauf dieser Arbeit ein Überblick über die Testsoftware HP QTP, wobei der Aufbau und die wichtigsten Funktionen genauer erläutert werden.

3.2. Das Hauptfenster

Nach dem Start von HP QTP erscheint das Hauptfenster. Im folgenden sollen die wichtigsten Elemente kurz erklärt werden.

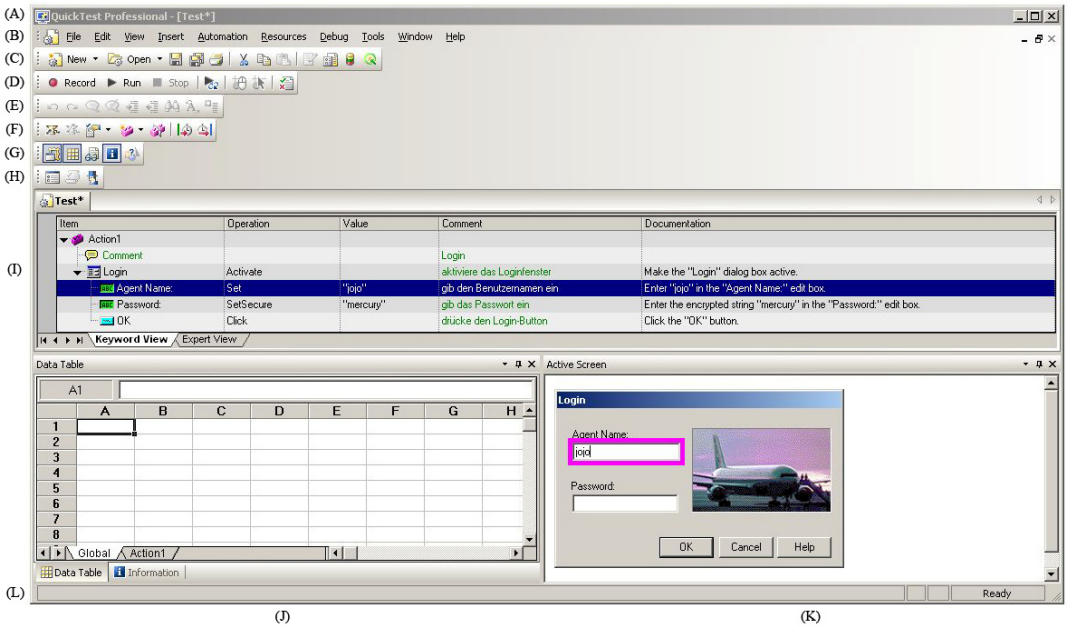

Abbildung 7 HP QTP-Hauptfenster

(A) die **Titelzeile**, die den Namen des geöffneten Test anzeigt

(B) die **Menüzeile** mit allen in HP QTP verfügbaren Funktionen

(C) die **Toolbar „Standard“** mit den normalen Windowsfunktionen

(D) die **Toolbar „Automation“** zum Aufzeichnen, Stoppen und Durchführen von Tests

(E) die **Toolbar „Debug“** mit Funktionen zum Debuggen von Tests

(F) die **Toolbar „Insert“** zum einfügen von Steps, Checkpoints, Actions, usw.

(G) die **Toolbar „View“** zum anzeigen verschiedener Fenster, die während eines Testprozesses hilfreich sein können

(H) die **Toolbar „Tools“** zur Kontrolle der Syntax, Nutzung des Object Spy und zur Einstellung der globalen Optionen

(I) das Testfenster mit dem **„Keyword-View“** und dem **„Epert-View“**

(J) die **„Data Table“** mit den globalen Datentabellen und zusätzlichen Tabellen für beliebig viele Actions oder Gruppen von Steps (mittels dieser Datentabellen kann der Test parametrisiert werden)

(K) der **„Active Screen“** mit dem Schnappschuss der zu testenden Anwendung im Moment der Aufzeichnung der Aktion (das aktive Element ist dabei hervorgehoben)

(L) die **Statuszeile** mit der Anzeige des aktuellen Status von HP QTP

Selbstverständlich ist die Größe der einzelnen Fenster und die Lage der Toolbars veränderbar und kann nach Belieben wieder auf die Standardwerte zurückgesetzt werden. [32 S.18]

3.2. Keyword View, Expert View und Test Results

HP QTP ermöglicht eine Automatisierung von Tests zum einen unter Verwendung des „Keyword View“ (die einzelnen Schritte werden tabellarisch in Form von Schlüsselwörtern beschrieben) und zum anderen unter Verwendung des „Expert View“ (die einzelnen Schritte werden in Skriptform beschrieben). Zwischen beiden Ansichten kann jederzeit umgeschaltet werden.

Wenn man durch eine Website oder Anwendung navigiert, zeichnet HP QTP jeden Schritt, der ausgeführt wurde auf. Diese Schritte werden in einer symbolbasierten Liste dargestellt, dem so genannten „Keyword View“.

Wenn der Test aufgezeichnet wurde, kann man mit HP QTP die spezifischen Eigenschaften eines jeden Objektes der Anwendung oder Website untersuchen. Außerdem können anschließend diese Schritte oder die Eigenschaften der Objekte bearbeitet bzw. neue Schritte hinzugefügt werden, um den Test zu optimieren. [32 S.8]

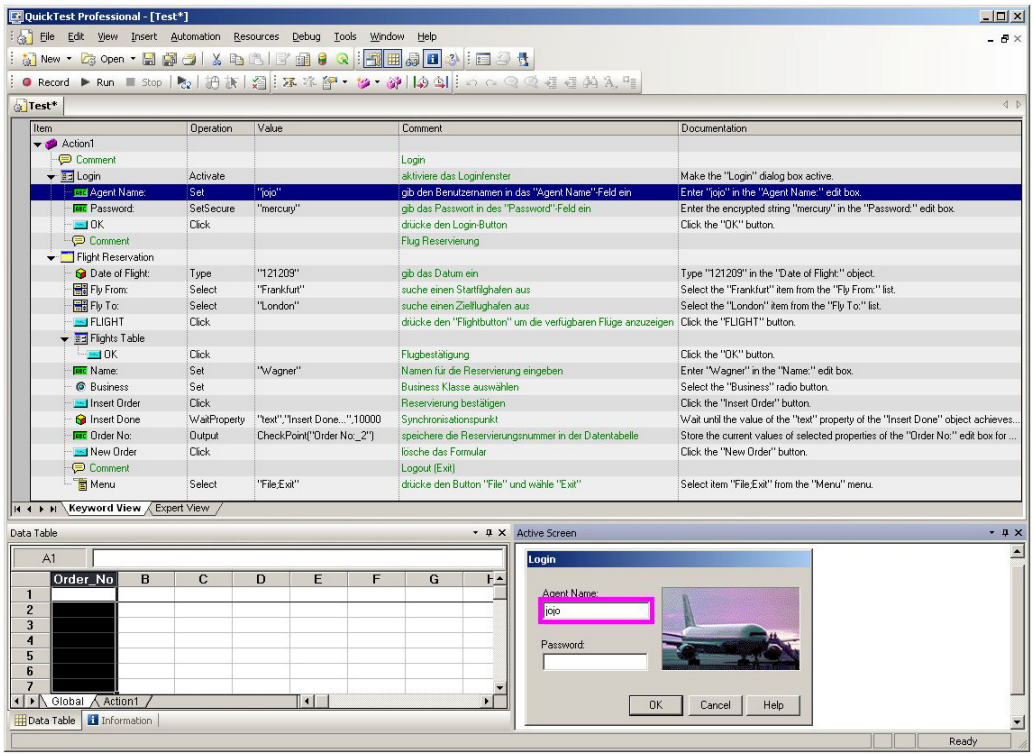

Abbildung 8 Darstellung der aufgezeichneten Schritte im Keyword View der HP QTP-Software

Im „Expert View“ wird der zur Action und deren Schritte korrespondierende Script-Code dargestellt. Der Wechsel zwischen beiden Ansichten erfolgt schnell und problemlos über die am unteren Bildrand befindlichen Reiter. [34 S.2-5]

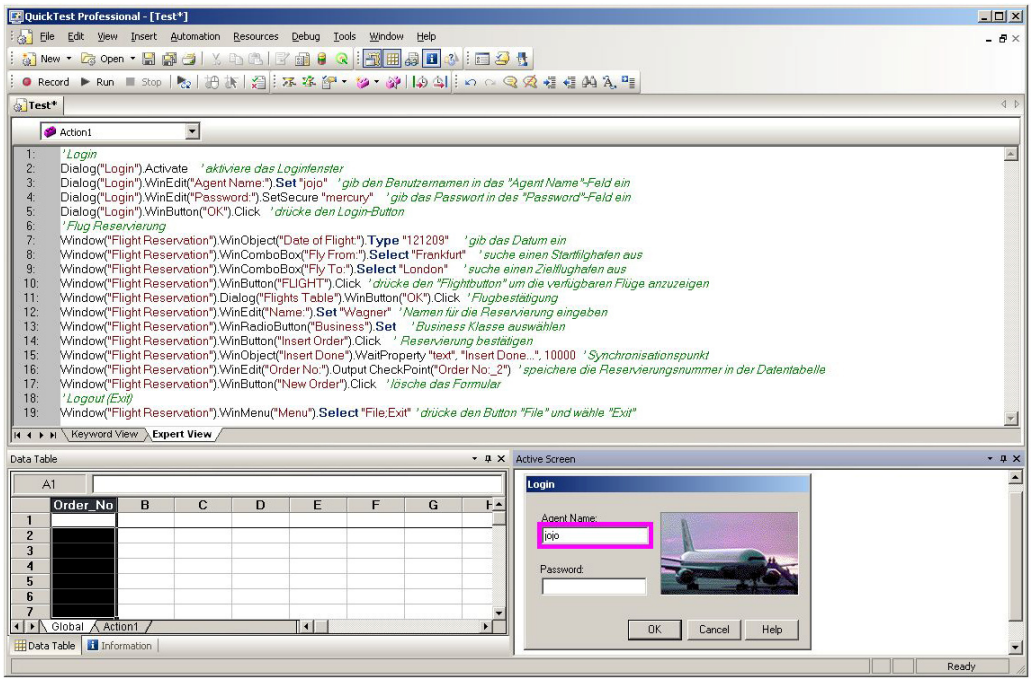

Abbildung 9 Darstellung der selben aufgezeichneten Schritte (siehe Abbildung oben) im Expert View

Im „Expert View“ wird VB-Script als reiner Programm-Quelltext dargestellt. Nach der Testdurchführung wird einen Report erzeugt, in dem detailliert angezeigt wird, welche Schritte im Test erfolgreich waren und welche Schritte fehlschlugen. [32 S.9]

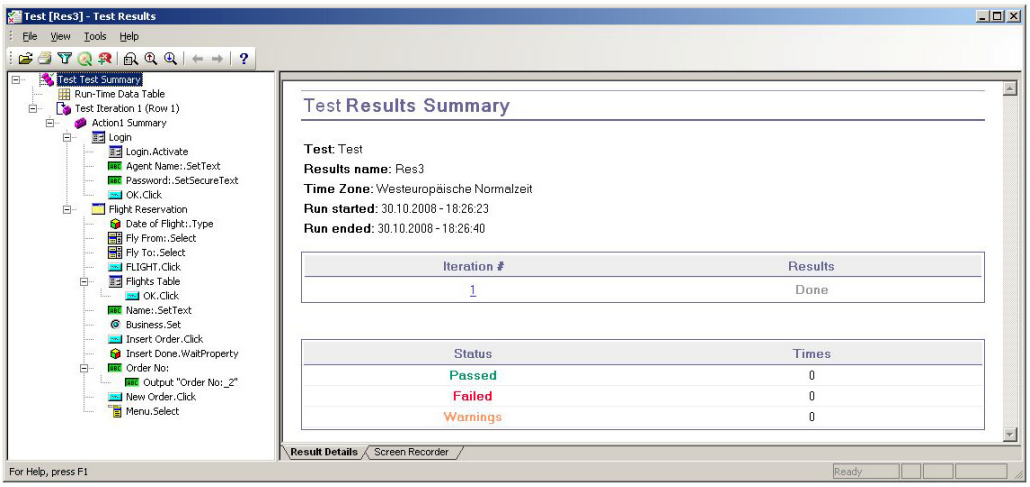

Abbildung 10 Testergebnis Überblick

3.3. Umgang mit Objekten

Der Begriff „Objekt“

Ein HP QTP Objekt ist ein grafisches Benutzerelement in einer Anwendung. Diese Objekte sind unterteilt in Klassen, wie beispielsweise Buttons, Fenster, Eingabefelder und viele mehr.

Durch HP QTP werden keine Objektinformation definiert. Es verwendet dieselben Informationen, die vom Anwendungsentwickler definierten wurden. [32 S.25]

Objekte und Klassen in HP QTP

Standardmäßig kann HP QTP Objekte und Klassen folgender Anwendungen/ Anwendungsgruppen erkennen:

- ActiveX-Komponenten in Webbrowser-Fenstern (Beispiel: Internet Explorer),
- Webbrowser Komponenten (Beispiel: Internet Explorer),
- Visual Basic Objekte,
- Windows Standard Anwendungen (Beispiel: WinZip Applikation). [31 S.8]

Standardobjekte in Windows- und Webapplikationen sind beispielsweise:

	Objekte Standard-Windows-Applikationen	Objekte Web-Applikationen
Objekte	▪ Fenster ▪ Dialogfelder	▪ Browser (-fenster) ▪ Webseite ▪ Bild
interaktive Objekte	▪ Buttons ▪ Checkboxen ▪ Editfelder ▪ Listboxen ▪ Radiobuttons	▪ Links ▪ Checkboxen ▪ Editfelder (URL) ▪ Listboxen (DropDown) ▪ Radiobuttons

Abbildung 11 Standardobjekte in Windows- und Webapplikationen [31 S.8]

Die Standardobjekte für Windows- und Webapplikationen sind sich sehr ähnlich, unterscheiden sich jedoch grundsätzlich in ihrer Ansprache über HP QTP.

```
' Das Programmfenster einer Standard-Windows-Applikation aktivieren.
Window("MyWindow").Activate
' Auf einen Button einer Standard-Windows-Applikation klicken.
Window("MyWindow").WinButton("MyButton").Click

' Synchronisation des Browserfensters einer Web-Applikation.
Browser("MyBrowser").Sync
' Auf ein Bild (Image) einer Web-Applikation klicken (z.B. Image als Link zum einloggen).
Browser("MyBrowser").Page("MyPage").Image("MyImage").Click
```

Listing 1 Ansprache der HP QTP-Objekte

Webobjekte innerhalb eines Browserfensters können nur dann erkannt werden, wenn das Plugin „Web“ beim Start von HP QTP mit geladen wurde. Bei geladenem Add-in können die anwendungsspezifischen Objekte (zum Beispiel in einem Browserfenster) entsprechend angesprochen werden wie vergleichbare Objekte in einer Standard-Windows-Applikation.

Identifizierung unterschiedlicher Objekte

Wie wird nun ein Objekt einer bestimmten Klasse von einem anderen Objekt der gleichen Klasse unterschieden?

In Abbildung 11 sind drei Buttons hervorgehoben. Sie gehören zur gleichen Klasse, haben die gleiche Größe, Farbe usw., unterscheiden sich aber in einem wesentlichen Punkt voneinander: der Beschriftung.

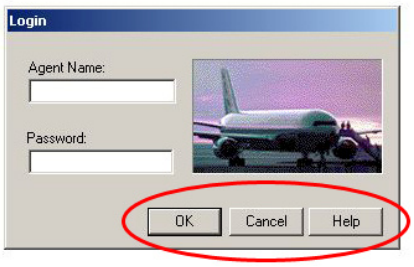

Abbildung 12 Drei Objekte (Buttons) der selben Klasse

Dieser Unterschied macht eine eindeutige Identifizierung der einzelnen Buttons möglich. Die einzelnen Objekte werden also anhand ihrer Objekteigenschaften („properties“) unterschieden.

Während des Aufzeichnens der Benutzereingaben „lernt“ HP QTP die Eigenschaften der verwendeten Objekte (beispielsweise Klasse, Text, Lage, Größe usw.) und sobald nur eine der Eigenschaften unterschiedlich ist, kann das Objekt eindeutig identifiziert werden. Je mehr Eigenschaften gelernt werden und je mehr Eigenschaften sich unterscheiden, umso leichter fällt diese Identifizierung.

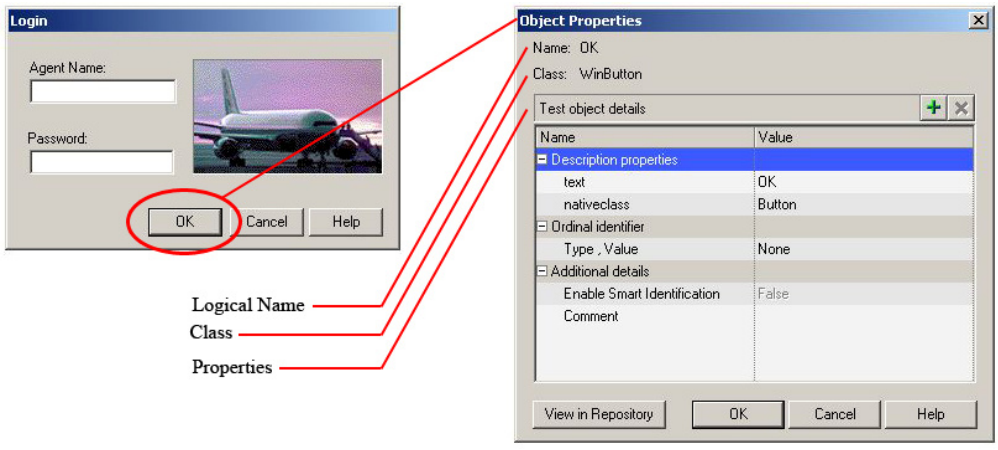

Abbildung 13 Eigenschaften des Objekts

HP QTP verwendet also zwei Methoden, um die Objekte eindeutig zu identifizieren:

- Erkennen des Objekt-Typs, in HP QTP „Class“ genannt, wie beispielsweise Fenster, Dialogboxen, Button usw..
- Lernen der Objekt-Eigenschaften, in HP QTP „Properties“ genannt, indem für jede Klasse von Objekten ein bestimmter Satz von Eigenschaften gespeichert wird, von denen meist schon ein kleiner Teil ausreicht, um das Objekt eindeutig zu identifizieren.

Nach dem Erlernen der Klasse und der Eigenschaften eines Objektes weist HP QTP dem Objekt einen eindeutigen Namen zu, genannt „Logical Name“. Dieser wird in HP QTP immer dann verwendet, wenn auf dieses Objekt verwiesen werden soll. Der „Logical Name“ kann verändert werden, um beispielsweise eine sinnvollere Bezeichnung zu wählen oder vorgegebene Namenskonventionen einzuhalten. [32 S.25]

Object Repository

Alle Objekte einer Anwendung werden im Object Repository gespeichert und verwaltet. Hier werden alle Objekte einer Anwendung gemeinsam mit ihren Eigenschaften angezeigt, die während einer Aufzeichnung benutz und hier gespeichert oder direkt zum Object Repository hinzugefügt wurden.

Eine Darstellung und wichtige Elemente des Object Repository sind in der folgenden Abbildung zu sehen.

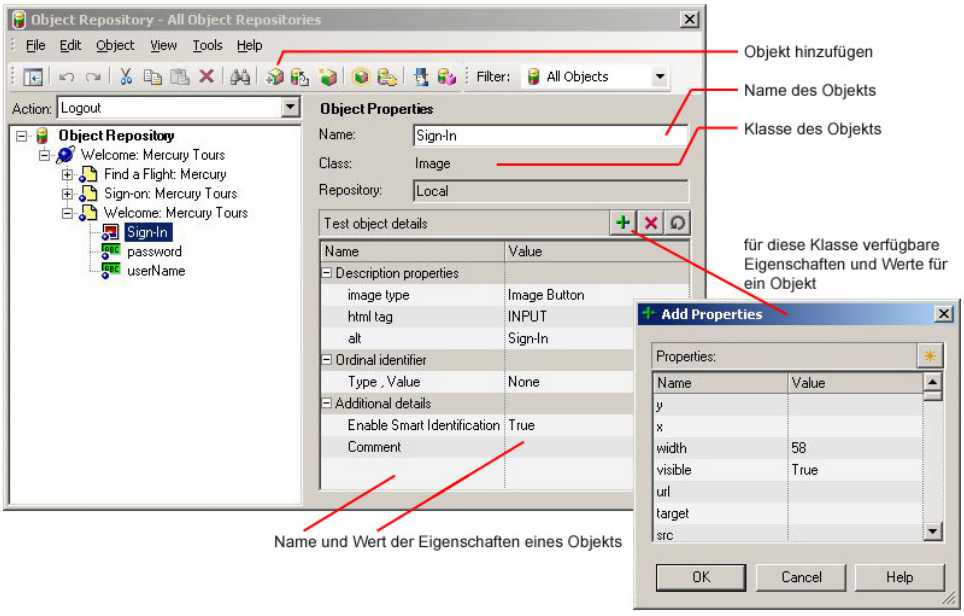

Abbildung 14 Object Repository

Als Beispiel für eine Web-Applikation wurde hier auf die Webseite von Mercury Tours zurückgegriffen. Das übergeordnete Objekt ist das Browserfenster „Welcome: Mercury Tours“. Alle darunter sichtbaren Objekte befinden sich im Kontext der Web-Anwendung und werden als untergeordnete Childobjekte dargestellt.
Somit hat das oben dargestellte Element „Sign-In“ die Klasse „Image“, „Welcome: Mercury Tours“ (Hauptfenster mit Darstellung des Webseiteninhaltes) die Klasse „Page“ und das Internet Explorer- Hauptfenster die Klasse „Browser“.
Für all diese Klassen sind eine Vielzahl von Eigenschaften und Werten verfügbar, die den Objekten zugeordnet werden können. [32 S.27]

Unterschiede zwischen Laufzeit-Objekten (Run-Time Objects) und Test-Objekten (Test Objects)

Alle Objekte einer Applikation, sowie die während der Aufzeichnung statischen Eigenschaftswerte dieser Objekte, werden als Test-Objekte im Object Repository gespeichert. Diese Test-Objekte werden während der Test-Ausführung verwendet, um die entsprechenden Laufzeit-Objekte zu finden.
Wurden die entsprechenden Objekte der Anwendung gefunden kann entschieden werden, ob die Eigenschaftswerte der Laufzeit-Objekte oder die Eigenschaftswerte der Test-Objekte verwendet werden sollen.
Test-Objekt-Eigenschaften enthalten nur eine Teilmenge der Eigenschaften eines Objekts und sind im Object Repository gespeichert. Diese werden zur Identifikation der Objekte benötigt.
Laufzeit-Objekt-Eigenschaften sind Eigenschaften eines Objekts, die in der Anwendung angezeigt werden und repräsentieren den kompletten Satz der Eigenschaften die für dieses Objekt.

Abfrage der Objekt-Eigenschaftswerte

Die Abfrage der Werte verschiedener Eigenschaften eines Objekts kann in sehr vielen Bereichen hilfreich sein.
Beispielsweise kann beim Debugging der Wert einer Eigenschaft überprüft werden, wenn vermutet wird, dass dieser falsch ist oder ein unerwünschtes Skript-Verhalten verursacht. In diesem Fall wird der Wert einer Eigenschaft im „Debug Viewer“ verfolgt bzw. abgefragt und kann für Vergleiche oder Berechnungen verwendet werden.

Auch bei speziellen Kontrollpunkten werden die abgefragten Werte verschiedener Eigenschaften benötigt, um überprüfen zu können, ob diese gültig sind oder nicht.
Des Weiteren können abgefragte Eigenschaftswerte in den Testbericht aufgenommen werden und dort vielseitig eingesetzt werden. Dadurch wird die Aufnahme von Ergebnissen und Bedingungen eines Tests in den Report ermöglicht.

3.4. Aufnahmemethoden

HP QTP kann gelegentlich bei der Erkennung aller Objekte, die während der Laufzeit registriert wurden, scheitern. Um Erfolgreich alle Gegenstände eines durchgeführten Test zu erkennen, kann es hilfreich sein die Aufnahmemethode zu ändern.
Ein Test in HP QTP kann in den folgenden Aufnahmemethoden durchgeführt werden:

- Normal,
- LowLevel,
- Analog.

Des Weiteren kann jede Kombination der verschieden Aufnahmemethoden in einem Test verwendet werden.

Der Normal-Aufnahmemodus (Normal Recording Mode)

HP QTP verwendet den Normal-Aufnahmemodus als Standardmodus zum aufzeichnen eines Test. In diesem Modus registriert QuickTest für jede Interaktion mit einem Objekt einen extra Schritt (Step). Der Normal-Aufnahmemodus ist für einen erfolgreichen Testablauf unabhängig von Fensterkoordinaten, Desktopkoordinaten oder der Bildschirmauflösung. Er ermöglicht HP QTP die Eigenschaften eines Objekts zu lernen und sie im „Object Repository“ für die Objektidentifizierung während der Laufzeit zu speichern. [33 S.14-3]

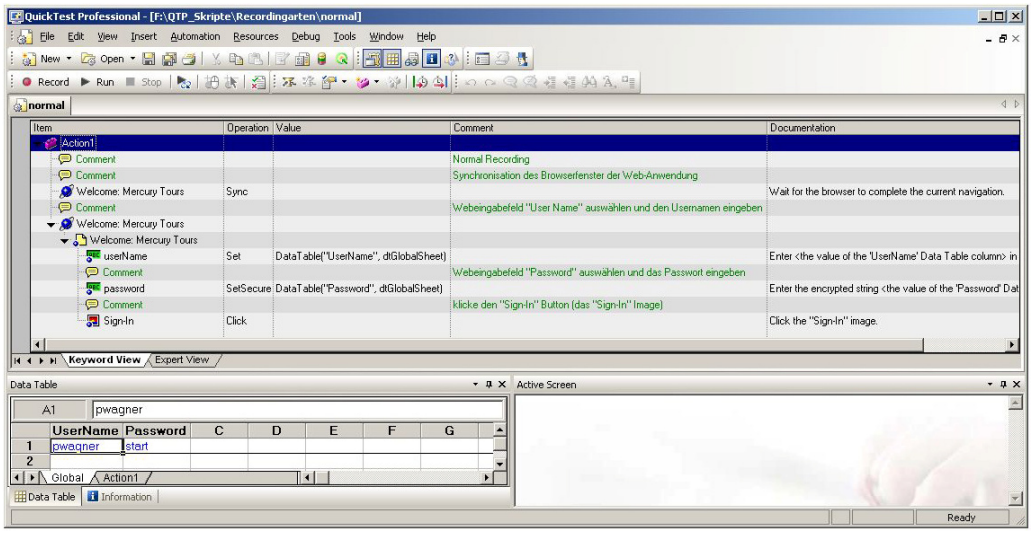

Abbildung 15 Normal Recording Mode im Keyword View

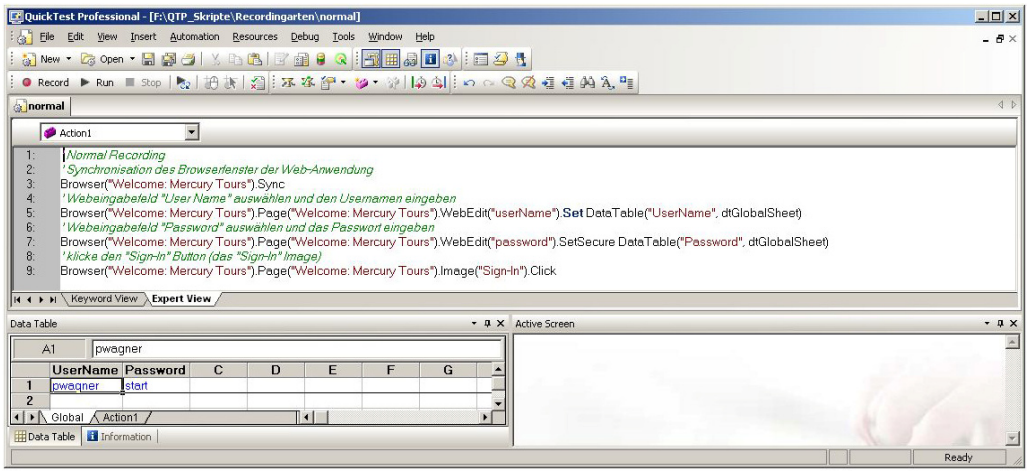

Abbildung 16 Normal Recording Mode im Expert View

Der Low-Level-Aufnahmemodus (Low-Level Recording Mode)

Der Low-Level-Aufnahmemodus ermöglicht es, ein Objekt einer Anwendung zu registrieren (aufzunehmen), unabhängig davon, ob HP QTP dieses Objekt kennt. Außerdem ermöglicht diese Methode auf ein Objekt in einer Anwendung zu klicken, selbst wenn HP QTP dieses nicht kennt. Des weiteren wird diese Methode verwendet, um genaue Koordinaten von Mausklicks oder Drag-and-Drop Operationen aufzuzeichnen, die für bestimmte Objekte erforderlich sind, um den Test erfolgreich durchführen zu können.

HP QTP:

- Lernt die Eigenschaften eines Laufzeitobjekts einer Windows-Klasse oder einer WinObject-Klasse und speichert diese Informationen im „Object Repository“ zur Objektidentifizierung während des Testlaufs,
- Registriert die genauen Koordinaten von Maus-Klicks, Drag-and-Drop Bewegungen und Eingabe-Operationen eines Objekts und
- Registriert einen Schritt für jede Interaktion mit einem Objekt

während der Low-Level-Aufnahme.

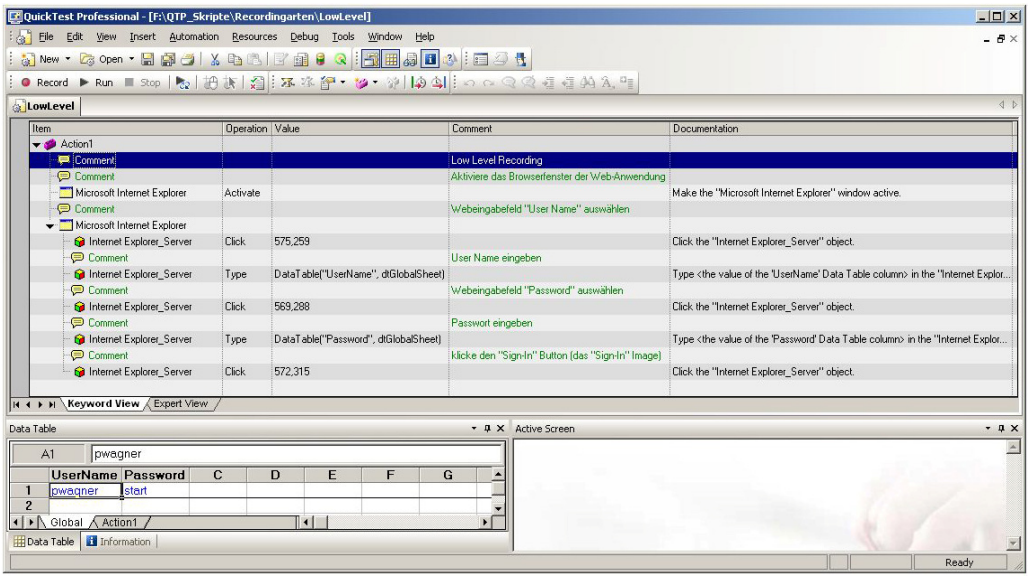

Abbildung 17 Low-Level Recording Mode im Keyword View

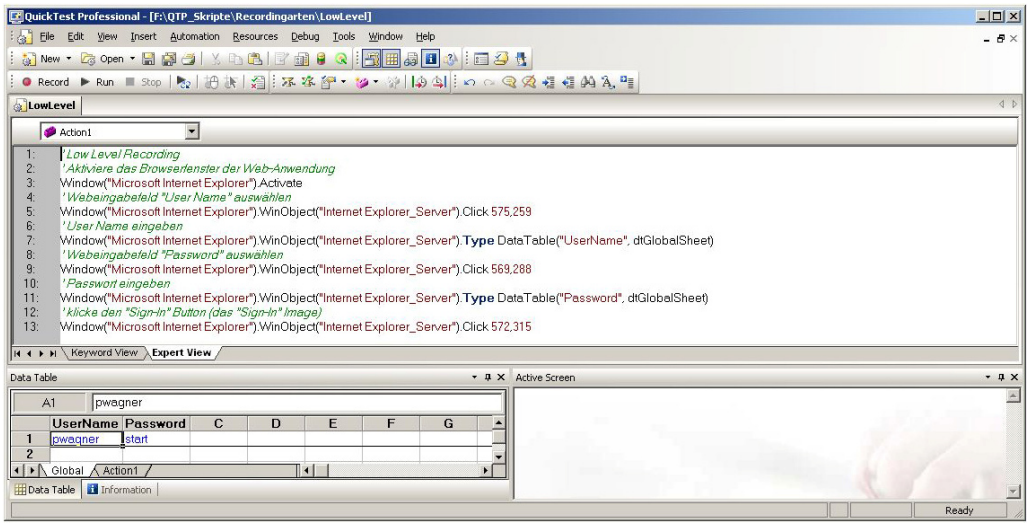

Abbildung 18 Low-Level Recording Mode im Expert View

Das Low-Level-Recording ist nützlich, wenn eine Anwendung von Maus-Klick oder Tastatur-Ereignissen abhängig ist und wird verwendet, wenn die genaue Position einer Operation auf dem Anwendungsbildschirm registriert werden muss.
Wird ein Objekt im Normal-Aufnahmemodus aufgenommen, führt HP QTP den Schritt auf das Objekt auch dann durch, wenn es sich auf dem Bildschirm zu einer anderen Position bewegt hat. Ist die Position des Objekts für den Test wichtig, muss in den Low-Level-Aufnahmemodus umgeschaltet werden, um es HP QTP zu ermöglichen das Objekt in Bezug auf seine x- und y-Koordinaten auf dem Bildschirm zu registrieren. Auf diese Weise wird der Testschritt nur erfolgreich gewertet, wenn das Objekt an der richtigen Stelle ist. [33 S.14-4, 14-5]

Der Analog-Aufnahmemodus (Analog Recording Mode)

Der Analog-Aufnahmemodus wird verwendet, um eine genaue Maus- oder Tastaturoperation, hinsichtlich der x-y-Koordinaten in einem Anwendungsfenster oder auf dem Desktop (Arbeitsfläche), durchzuführen. Dieser Modus ist nützlich, um Operationen durchzuführen die nicht als Objekte gespeichert werden können. Beispielsweise ist das Zeichnen eines Bildes in der Microsoft Paint Applikation oder die Aufnahme einer digitalen Signatur (siehe Beispielapplikation Flight Reservation – „Fax Order…“) ein klarer Fall für das Analog-Recording. Mit den beiden anderen Aufnahmemodi wäre dies nicht möglich. [33 S.14-6]

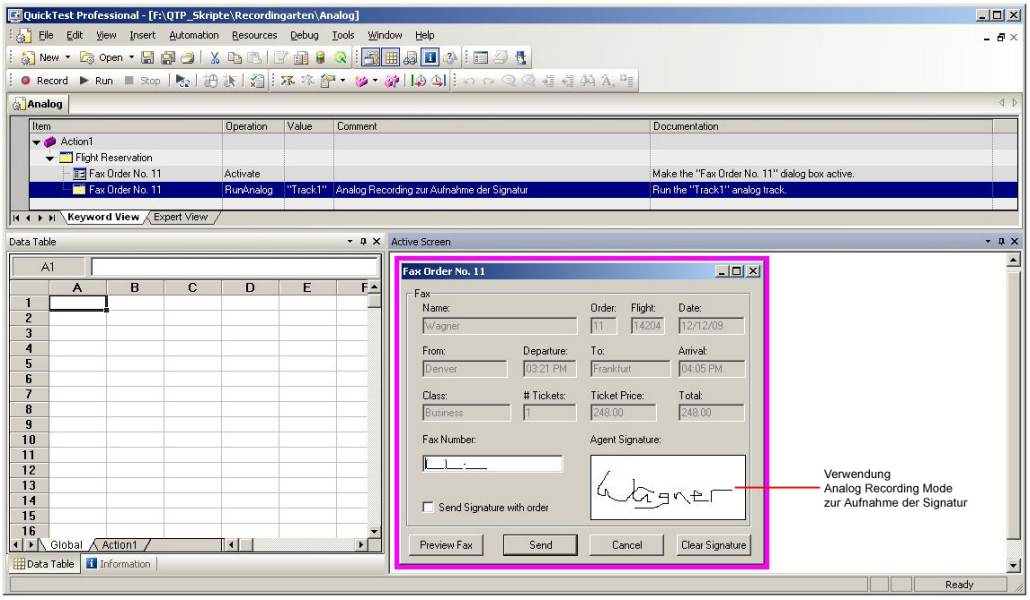

Abbildung 19 Analog Recording Mode im Keyword View zur Aufnahme der Signatur

3.5. Virtuelle Objekte

Man kann HP QTP beibringen, jedes Gebiet einer Anwendung als ein Objekt anzuerkennen, in dem man es als virtuelles Objekt definiert. Virtuelle Objekte ermöglichen es, Tests auf Objekte aufzuzeichnen und durchzuführen, die durch HP QTP normalerweise nicht erkannt werden.

Objekte die normalerweise durch HP QTP nicht erkannt werden, können demnach als virtuelles Objekt definiert werden und zu den Standardklassen, beispielsweise als Button oder Checkbox, hinzugefügt werden. Man kann aber nur Objekte als virtuelle Objekte definieren, auf denen HP QTP „Klick“ oder „Doppelklick“-Methoden registrieren kann, ansonsten wird das virtuelle Objekt ignoriert.

Um Tests richtig aufzuzeichnen und durchzuführen muss sichergestellt werden, dass Webseiten- oder Anwendungsfenster die selbe Größe besitzen und sich in der selben Position befinden, wie zu dem Zeitpunkt, zu dem die virtuellen Objekte definiert worden sind. Des weiteren dürfen Grafiken, Images, Buttons oder ähnliche Elemente, auf die eine „Klick“ oder „Doppelklick“-Methode ausgeführt werden soll, nach ihrer Definition als virtuelles Objekt nicht mehr verschoben werden.

HP QTP sucht nicht im Webseiten- oder Anwendungsfenster den während der Definition markierten Bereich, sondern führt in diesem Bereich nur die „Klick“ oder „Doppelklick“-Methode aus. Wurde beispielsweise ein Button aus diesem Bereich verschoben, kann die Methode nicht ausgeführt werden und der Test schlägt fehl. Es findet also keine grafikorientierte Objekterkennung statt.

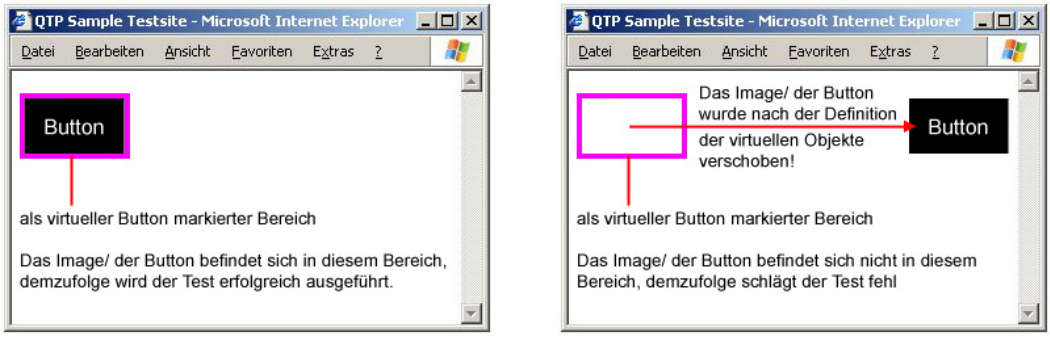

Abbildung 20 Vergleich eines erfolgreichen und eines fehlgeschlagen Test mit virtuellen Objekten

Virtuelle Objekte können darüber hinaus nicht mit einem HP QTP-Test zusammen gespeichert werden. Ein Test, der virtuelle Objekte verwendet, muss an dem Computer durchgeführt werden, der die Virtuelle-Objektkollektion enthält. Um die Virtuelle-Objektkollektion an einem anderen Rechner nutzen zu können müssen die Daten aus

dem VoTamplate-Ordner (Speicherort der Virtuelle-Objektkollektion) kopiert werden. [33 S.14-12]

3.6. Screenshot

Unter einem Screenshot versteht man das Abspeichern oder die Ausgabe des aktuellen grafischen Bildschirminhaltes eines PCs. Mit der CaptureBitmap-Methode ist es möglich, HP QTP einen Screenshot eines Objekts erzeugen zu lassen und diesen als .png oder .bmp Image abzuspeichern. Dabei kann entschieden werden, ob der gesamte Desktop (alles Sichtbare auf dem Display), einzelne Fenster (z.B. das Browserfenster) oder ausgewählte Objekte (z.B. ein Button) erfasst und gespeichert werden sollen.

Die Syntax der CaptureBitmap-Methode sieht folgender maßen aus [35]:

```
object.CaptureBitmap FullFileName, [OverrideExisting]
```

Listing 2 Syntax der CaptureBitmap-Methode

Das Argument `object` ist ein Objekt vom Typ Browser oder WinObject, also ein Objekt einer Webapplikation oder Windowsapplikation. `FullFileName` ist ein erforderliches Attribut, welches, wie der Name schon sagt, den vollen Dateinamen enthält. Hierbei muss der komplette Pfadname und der dazugehörige Dateityp (.png oder .bmp) des Images angegeben werden. Wird ein relativer Pfad mitgeteilt, so wird das Image automatisch zum Testergebnisordner bei jedem Testlauf hinzugefügt. Das `OverrideExisting`-Attribut ist optional und besitzt einen Boolean-Wert. Damit kann entschieden werden, ob ein erfasstest Objekt, das sich schon im angegebenen Pfad (Ordner) befindet, überschrieben werden soll oder nicht. Die beiden möglichen Werte sind „True“ oder „False“. [35]

Dazu einige Beispiele:

```
' Screenshot vom gesamten Browserfenster
Browser("MyBrowser").CaptureBitmap "c:\browser.bmp", True
' Screenshot nur von der Webseite
Browser("MyBrowser ").Page("MyPage").CaptureBitmap "c:\page.bmp", True
' Screenshot nur von einem Button
Browser("MyBrowser ").Page("MyPage").Image("MyButton").CaptureBitmap "c:\button.bmp", True
```

Listing 3 Screenshot in HP QTP

Zu beachten ist bei der CaptureBitmap-Methode, das die Objekte, von den ein Screenshot erzeugt werden soll, im Display sichtbar sind. Ist nur ein Teil sichtbar, wird auch nur der sichtbare Teil des Objekts gespeichert. Des Weiteren ist zu beachten, das die in der benutzten Methode ausgewählten Objekte im Object Repository bekannt sind. Sind diese nicht bekannt muss die programmatische Beschreibung verwendet werden.

Beispiel, Screenshot mit programmatischer Beschreibung:

```
' Screenshot nur von einem Button mit der programmatischen Beschreibung
Browser("MyBrowser").Page("MyPage").Image("name:=MyButton").CaptureBitmap "c:\button.bmp", True
```

Listing 4 Screenshot mit programmatischer Beschreibung

4. Grafikbasierende Objekterkennung mittels HP QuickTest Professional

4.1. Einführung

Zur Sicherung der Software-Qualität und zur Automatisierung bzw. Durchführung von Funktions- und Regressionstests sollte die Testsoftware fehlerfrei und stabil arbeiten. Wie bereits beschrieben verwendet HP QTP zur Erstellung der Testfälle die sogenannte Objekterkennung, mit deren Hilfe die Elemente direkt aus dem Applikationsbildschirm erfasst werden. Diese Funktion von HP QTP ist sehr wichtig und soll im weiteren Verlauf dieser Arbeit genauer untersucht werden, um mögliche Fehlfunktionen zu erkennen. Im Anschluss kann nach der umfangreichen Analyse ein möglicher Lösungsansatz abgeleitet und die Lösungsstrategie definiert werden.

4.2. Fehlerfreie Objekterkennung

Bevor mit der Aufzeichnung begonnen wird, sollten die folgenden Punkte nochmals überprüft werden:

- Sicher stellen, dass die zu testende Anwendung für den Test bereit ist und dass sie sich in einem definierten Ausgangszustand befindet (sind eventuell irgendwelche Warnfenster erschienen, ist die Anwendung vielleicht minimiert).
- Testschritte probehalber ohne Aufzeichnung ausführen, um zu überprüfen, ob die Anwendung erwartungsgemäß reagiert.
- Überprüfen der Testdaten, ob diese sinnvoll und gültig sind (Zahlenformate, Datumsangaben, etc.).
- Überprüfen der Testumgebung (sind notwendige Netzlaufwerke verbunden, laufen benötigte Server fehlerfrei).
- Prüfen, ob alle Voraussetzungen in HP QTP gegeben sind (alle notwendigen Add-ins und ein neuer, leerer Test geladen).

Wurden alle Punkte überprüft, kann mit der Aufzeichnung des Tests begonnen werden.

Vorbereitung der Testumgebung und Aufzeichnen des Tests

HP QTP muss beim Start der zu testenden Anwendung geöffnet sein. Dazu wird zu Beginn HP QTP gestartet.

Als erstes erscheint der sogenannte „Add-in Manager“. Je nach Lizenz bzw. installierten Add-ins werden hier mehr oder weniger Add-ins angezeigt. Da die erfolgreiche Objekterkennung anhand eines Logins bei einer Webseite gezeigt werden soll, muss das „Web“ Add-in aktiviert werden, damit alle Objekte erkannt werden können. Generell sollten nur notwendige Add-ins aktiviert werden, da dadurch die Performance und die Erkennungsrate von Objekten steigt.

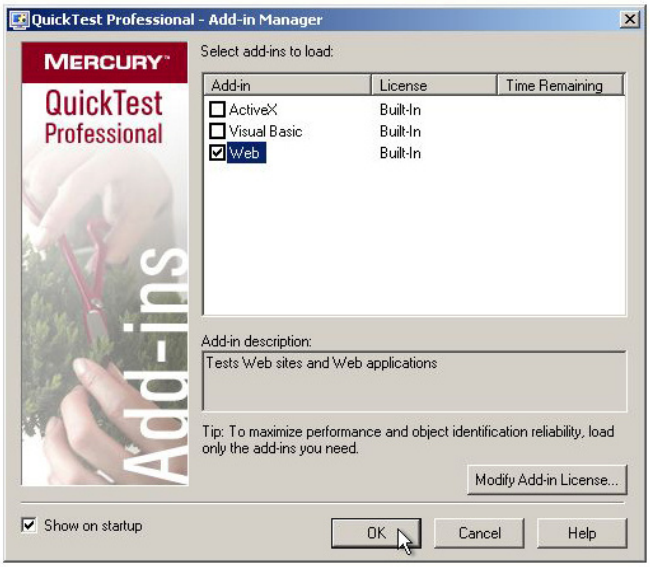

Abbildung 21 Add-in Manager mit aktiviertem Web Add-in

Durch einen Klick auf den „OK“-Button wird das Fenster geschlossen und HP QTP wird gestartet. HP QTP stellt zu Beginn immer einen neuen (leeren) Test zur Verfügung.

Im nächsten Schritt sollte die zu testende Anwendung gestartet werden und in einen definierten Initialzustand versetzt werden. Dazu wird die simple Testapplikation „Mercury Tours“ Webseite im Browser geladen.

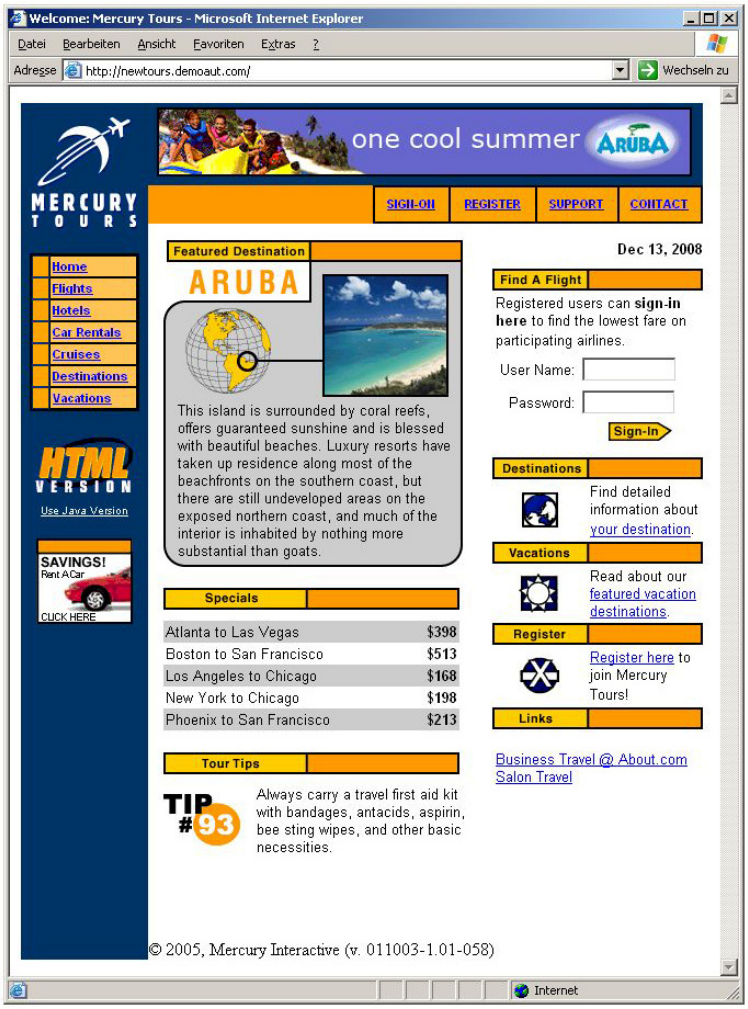

Abbildung 22 Webseite der Testapplikation „Mercury Tours“

Damit ist die Vorbereitung abgeschlossen und der Test kann aufgezeichnet werden. Dazu wird in das HP QTP-Hauptfenster gewechselt und die Aufzeichnung über Menü „Automation“ → „Record“ oder besser direkt über den Schalter „Record“ in der Symbolleiste „Automation“ gestartet. Danach erscheint das Fenster „Record and Run Settings“, in dem die Option „Record and run test on any open browser“ im Tab „Web“ ausgewählt wird. Durch einen Klick auf den „OK“-Button wird die Aufzeichnung gestartet. Während der Aufzeichnung ändert sich die Größe des HP QTP-Fensters und das Fenster der zu testenden Anwendung wird sichtbar. Unabhängig davon kann man dies auch über die Windows-Taskbar erreichen.

Klickt man nun in der „Mercury Tours“ Webseite in das Eingabefeld „User Name“ und gibt dort einen Benutzernamen ein, wird dies von HP QTP registriert und ein neuer Schritt (Step) im Test erzeugt. Dies gelingt nur, da das Eingabefeld ein Standartobjekt einer Webapplikation ist, welche HP QTP erkennen und aufzeichnen kann. Darüber hinaus wird das Objekt mit dem dazugehörigen Namen, der Klasse und den

Eigenschaften im „Object Repository“ abgespeichert. In diesem Beispiel wurden gleichzeitig zwei Objekte der Klasse Browser (das Internet Explorer-Hauptfenster) und der Klasse Page (Hauptseite mit Darstellung des Webseiteninhalts) erkannt. Diese sind wichtig, damit das Eingabefeld später beim Test-Run gefunden werden kann. Das Eingabefeld befindet sich im Kontext der Webapplikation und ist ein untergeordnetes Childobjket des Browser- und Page-Objekts.

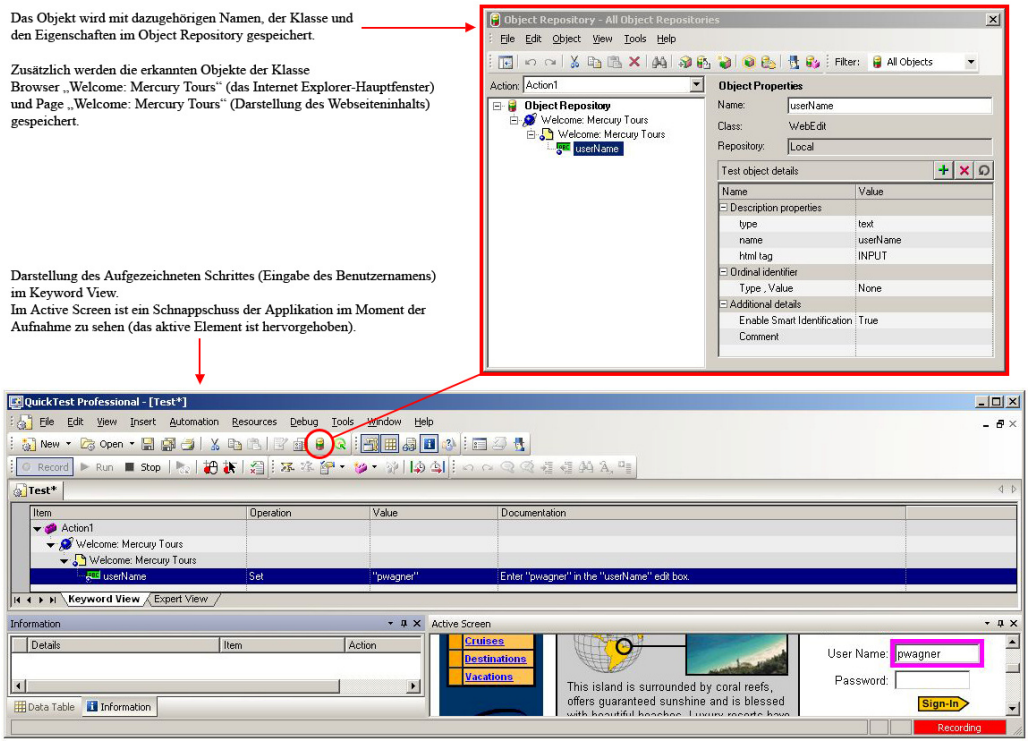

Abbildung 23 Darstellung des aufgezeichneten Schrittes (Eingabe des Benutzernamens) in HP QTP

Anschließend wird dieselbe Prozedur mit dem Eingabefeld „Password“ und dem Image „Sign-In“ (das Image stellt den Login-Button dar) vollzogen. Jeder aufgezeichnete Schritt und jedes Objekt wird dabei in einer neuen Zeile dargestellt und durch verschiedene Symbole gekennzeichnet (siehe folgende Abbildung).

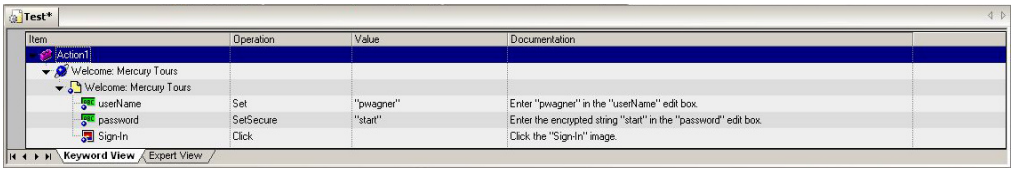

Abbildung 24 Die gesamten aufgezeichneten Schritte im Keyword View

Der korrespondierende Script-Code der einzelnen Schritte wird im Expert View dargestellt und kann problemlos über den am unteren Bildrand befindlichen Reiter angesehen werden.

Abbildung 25 Korrespondierender Script-Code der einzelnen Schritte im Expert View

Damit ist das Beispielscript erfasst und die Aufzeichnung kann beendet werden, indem in das HP QTP-Fenster gewechselt und der „Stop“-Schalter angeklickt wird.

Test durchführen

Die Testdurchführung wird entweder durch Drücken des Schalters „Run“ direkt im HP QTP-Hauptfenster (bei Fehlersuche und Optimierung) oder später bei der eigentlichen Testdurchführung durch den Tester aus dem HP QC heraus im Rahmen eines Testsets gestartet.
Drückt man in der Symbolleiste von HP QTP auf den Schalter „Run“, beginnt, nach einer kurzen Abfrage über den Speicherort der Testergebnisse, die automatische Durchführung der vorher aufgezeichneten Schritte.

Test auswerten

Ist die Testausführung beendet, erscheint das folgende Fenster „Test Results“:

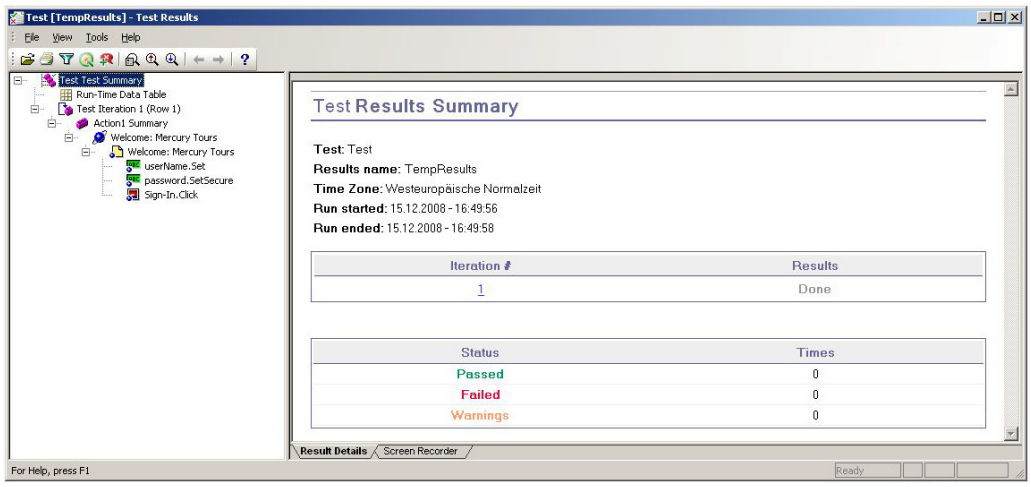

Abbildung 26 Testergebnisse (Zusammenfassung)

Auf der linken Seite ist eine Baumstruktur mit den Testergebnissen als symbolische Liste abgebildet. Jeder ausgeführte Schritt wird hier noch einmal dargestellt. Auf der rechten Seite befinden sich die detaillierten Testergebnisse. Die obere Tabelle gibt an, wie oft der Test durchlaufen wurde (je Interaktion eine Zeile) und zeigt unter „Results“ das jeweilige Ergebnis des Durchlaufs an. In der unteren Tabelle wird die Anzahl der Checkpoints und Reports angezeigt, die erfolgreich, fehlerhaft oder mit Warnungen durchlaufen wurden. Darüber hinaus kann links ein Schritt markiert und somit auf der rechten Seite eine detaillierte Beschreibung des Steps angezeigt werden.
Um zu verdeutlichen, dass wirklich alle Objekte erkannt und die einzelnen Schritte erfolgreich ausgeführt wurden, muss das Script etwas modifiziert werden.

Das Script sieht vor der Modifikation so aus:

```
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("userName").Set "pwagner"
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("password").SetSecure "start"
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").Image("Sign-In").Click
```

Listing 5 Testscript zur fehlerfreien Objekterkennung vor der Modifikation

Im Folgenden soll nach der Eingabe des Benutzernamens und des Passwortes überprüft werden, ob das Objekt erkannt wurde und die richtigen Testdaten in die Eingabefelder eingetragen wurden. Dazu wird nach der Eingabe der aktuelle Wert des Objekts mit der `GetROProperty`-Funktion abgefragt, mit dem zu erwartenden Wert verglichen und danach in einen Report-Eintrag gespeichert, der später bei den „Test Results“ ausgelesen werden kann. Des Weiteren soll der Imagename des Login-Buttons abgefragt werden und nur gedrückt werden, wenn dieser mit dem Erwartungswert übereinstimmt. Andernfalls soll der Test abgebrochen werden. Je nach dem, ob der Loginversuch (also das Drücken des Buttons) erfolgreich war oder nicht, wird ein Report mit dem entsprechenden Ergebnis erzeugt.

Das Script sieht nach der Modifikation so aus:

```
' Variablendeklaration
Dim wert                    ' leere Variable – später zur Aufnahme des abgefragten Wertes des Objekts
Dim userName                ' Variable zum Vergleich des Benutzernamens
userName = "pwagner"        ' Erwartungswert des Benutzernamens
Dim password                ' Variable zum Vergleich des Passwortes
password = "start"          ' Erwartungswert des Passwortes
Dim name_image              ' Variable zum Vergleich des Image Namen (Sign-In bzw. Login-Button)
```

```
name_image = "login"              ' Erwartungswert des Image Namen
' Web-Edit Objekt => userName => Eingabe des Benutzernamens
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("userName").Set "pwagner"
' Abfrage: Web-Edit Objekt – welcher Benutzername wurde eingegeben?
wert = Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("userName").GetROProperty("value")
' prüfe, ob der Benutzername dem Erwartungswert entspricht und entscheide entsprechend der aktuellen Situation
If (wert = userName) Then
   ' erzeuge einen Report-Eintrag mit „Passed“ (Erfolgreich), wenn der Benutzername dem Erwartungswert entspricht
   Reporter.ReportEvent micPass, "Objekt erkannt!", "Objekt erkannt, der richtige Benutzername wurde eingegeben!"
   Else
   ' erzeuge einen Report-Eintrag mit „Failed“ (nicht Erfolgreich), wenn der Benutzername dem Erwartungswert nicht entspricht
   Reporter.ReportEvent micFail, "Fehler!", "Ein Fehler ist aufgetreten!"
   ' beende den „Run“
   ExitRun (1)
End If

' Web-Edit Objekt => password => Eingabe des Passwortes
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("password").SetSecure "start"
' Abfrage: Web-Edit Objekt – welches Passwort wurde eingegeben?
wert = Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").WebEdit("password").GetROProperty("value")
' prüfe, ob das Passwort dem Erwartungswert entspricht und entscheide entsprechend der aktuellen Situation
If (wert = password) Then
   ' erzeuge einen Report-Eintrag mit „Passed“ (Erfolgreich), wenn das Passwort dem Erwartungswert entspricht
   Reporter.ReportEvent micPass, "Objekt erkannt!", "Objekt erkannt, das richtige Passwort wurde eingegeben!"
   Else
   ' erzeuge einen Report-Eintrag mit „Failed“ (nicht Erfolgreich), wenn das Passwort dem Erwartungswert nicht entspricht
   Reporter.ReportEvent micFail, "Fehler!", "Ein Fehler ist aufgetreten!"
   ' beende den „Run“
   ExitRun (2)
End If

' Image Objekt  => Sign-In (Grafik als Login-Button)
' Abfrage: Image Objekt – welchen Namen hat das Image?
wert = Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").Image("Sign-In").GetROProperty("name")
' prüfe, ob der Name des Image Objekts dem Erwartungswert entspricht und entscheide entsprechend der aktuellen Situation
If (wert = name_image) Then
   ' der Name des Images entspricht dem Erwartungswert – drück den Button
   Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").Image("Sign-In").Click
   ' erzeuge einen Report-Eintrag mit „Passed“ (Erfolgreich)
   Reporter.ReportEvent micPass, "Login", "Der Loginversuch war erfolgreich!"
   Else
   ' der Name des Images entspricht nicht dem Erwartungswert – erzeuge einen Report-Eintrag mit „Failed“ (nicht Erfolgreich)
   Reporter.ReportEvent micFail, "Login", "Der Loginversuch war nicht erfolgreich!"
   ' beende den „Run“
   ExitRun (3)
End If
```

Listing 6 Testscript zur fehlerfreien Objekterkennung nach der Modifikation

Wird nun der Test durchgeführt, erhält man folgende Testergebnisse:

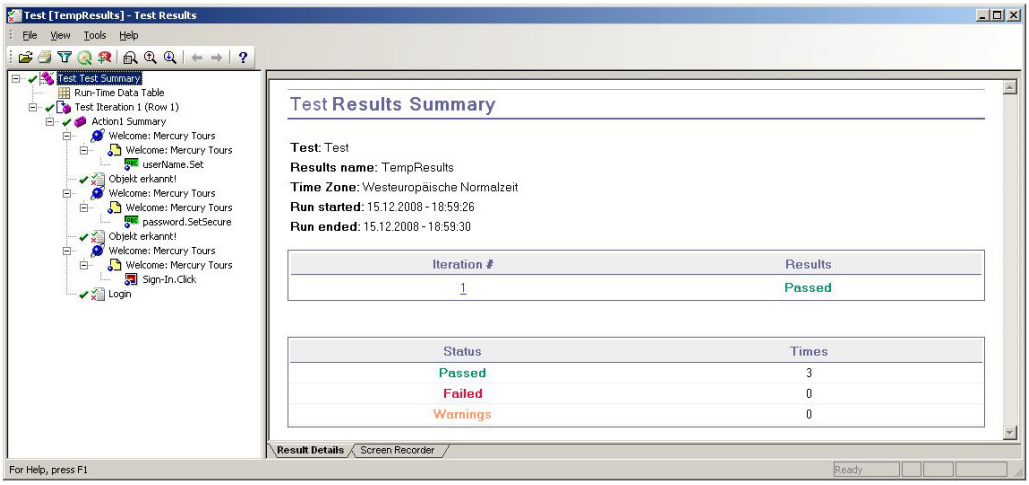

Abbildung 27 Testergebnisse (Zusammenfassung) nach Durchführung des modifizierten Tests

Wie man in dieser Abbildung sehr schön erkennen kann, wurden alle Objekte erkannt, alle Schritte (Steps) wurden ausgeführt und als erfolgreich gewertet („Passed“). Die detaillierten Angaben der einzelnen Reports verdeutlichen dies noch mehr:

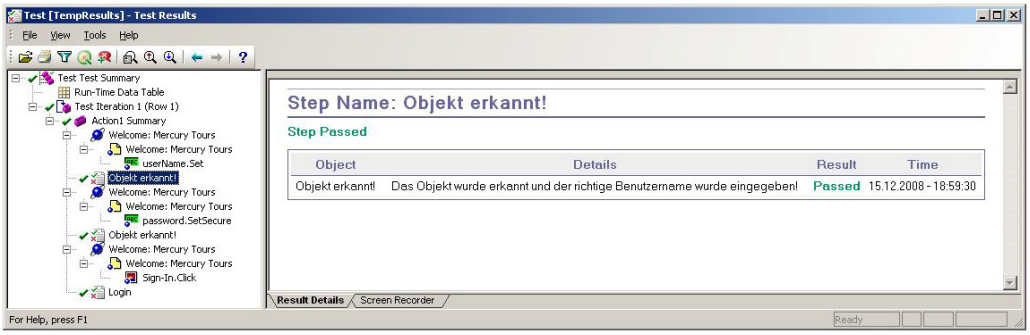

Abbildung 28 Reporteintrag - Benutzername

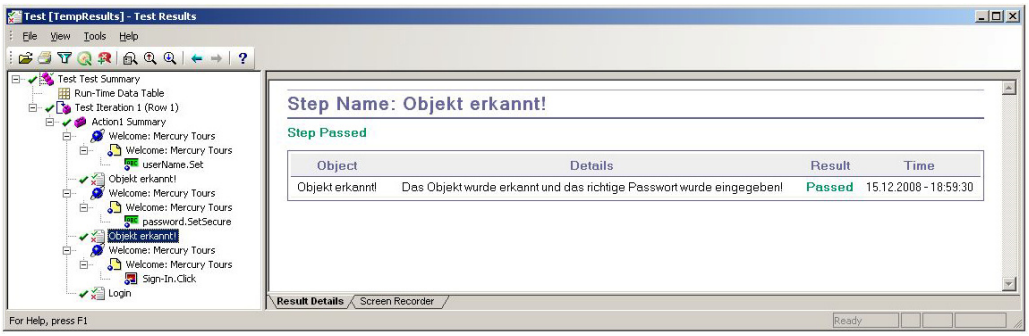

Abbildung 29 Reporteintrag - Passwort

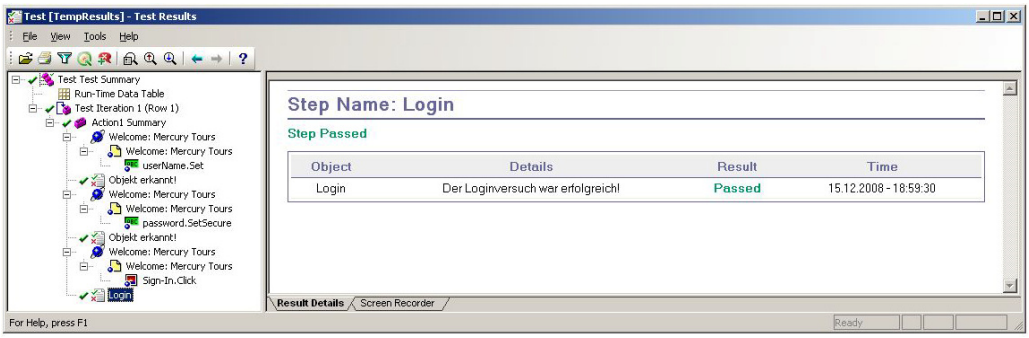

Abbildung 30 Reporteintrag – Imagename bzw. Login erfolgreich

4.3. Fehlerhafte Objekterkennung

Immer wenn Recordingaktionen durchgeführt werden, fügt HP QTP automatisch die entsprechenden, zur Erkennung der verwendeten Objekte, erforderlichen Einträge in das Object Repository ein. Diese Objekterkennung funktioniert allerdings nicht immer fehlerfrei. Es gelingt nicht immer, alle Objekte korrekt zu identifizieren, was zu Problemen führen kann. Auslöser könnte beispielsweise eine unsaubere Programmierung des Testobjekts sein. Fehler im Quelltext oder eine unzureichende Beschreibungen des Objekts können zu Verwechslungen führen. Fehlende HP QTP-Plugins oder Anwendungen, die asynchron zum Anwender im Hintergrund arbeiten (z.B. AJAX-Webanwendungen), könnten ebenfalls verantwortlich sein. Darüber hinaus sind Objekte einer Anwendung dynamischen Veränderungen unterworfen. Oft ist die genaue Position des Objekts nicht bekannt. Einige Objekte können vom Anwender verschoben oder ausgeblendet werden, andere sind abhängig von konkreten Daten oder tauchen erst nach einer bestimmten Wartezeit auf.

Bemerkbar machen sich diese Fehler auf unterschiedliche Weise. Einige Objekte können beispielsweise beim Aufzeichnen nicht aufgenommen werden, weil diese HP QTP gänzlich unbekannt sind. Erkannte Objekte bzw. deren Eigenschaften entsprechen nicht dem, was der Anwender sieht oder die Anzeigedauer von Objekten wird nicht berücksichtigt (z.B. ein Ladebalken verschwindet für einige Millisekunden).

Zeigen des Problems

Das folgende Beispiel soll zeigen, das HP QTP nicht immer in der Lage ist Objekte richtig zu erkennen. Ein Test könnte beispielsweise die Aufgabe besitzen im Browser zur Webseite von Google Maps zu navigieren. Dort soll ein Ort gesucht werden und zur

Anzeige gebracht werden. Danach soll mittels des „Plus“ Buttons in den abgebildeten Kartenabschnitt hinein gezoomt werden.

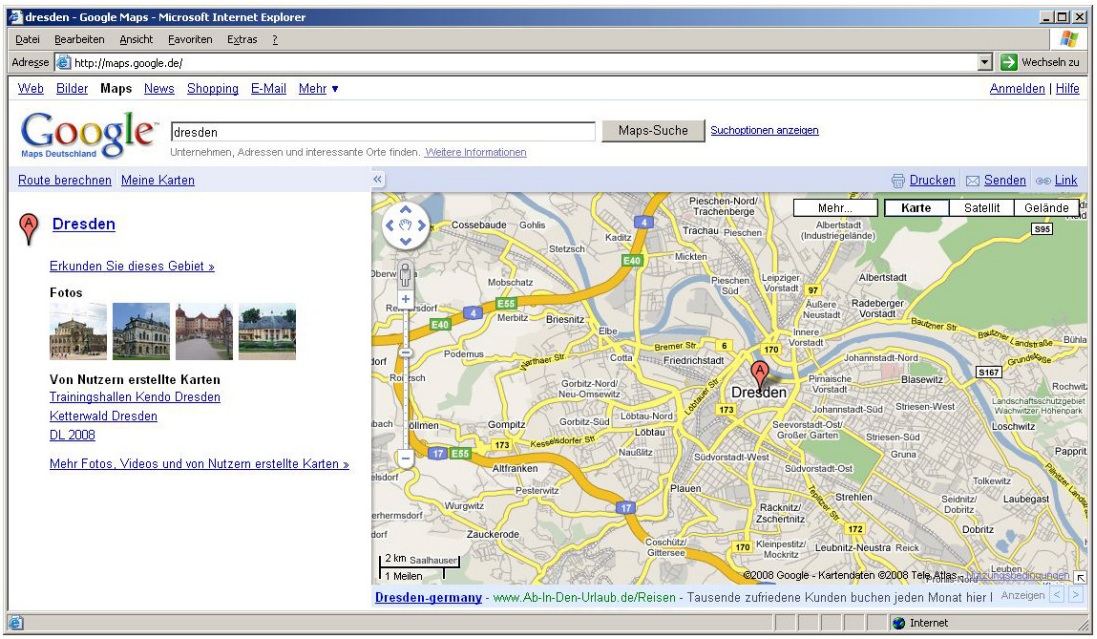

Abbildung 31 Beispielwebseite von Google Maps

Als erstes wird nun wieder HP QTP mit dem „Web“ Add-in gestartet und ein neuer (leerer) Test geladen. Danach wird der Browser bzw. die Webseite in den gewünschten Ausgangszustand gebracht. Wird nun wieder der „Record“ Schalter betätig kann die Aufzeichnung beginnen. HP QTP ist in der Lage das Eingabefeld zu erkennen und registriert ebenfalls, welcher Ort gesucht werden soll. Auch das Drücken des „Maps-Suche“ Buttons wird erfasst und die verlangte Funktion wird ausgeführt.

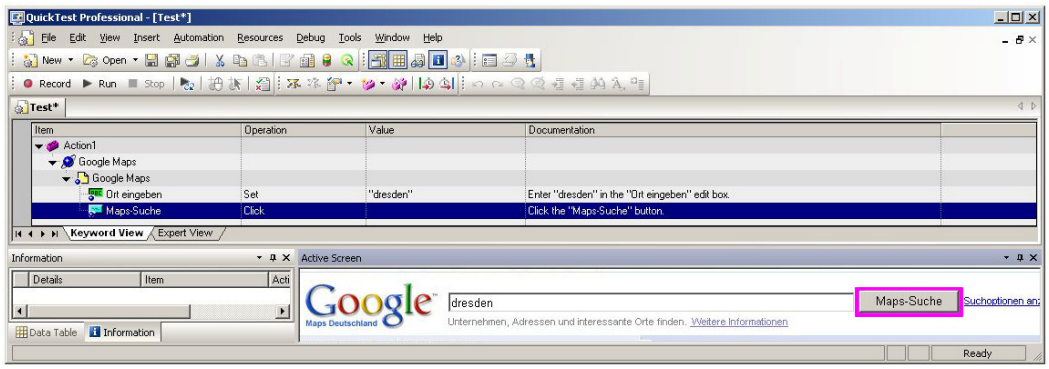

Abbildung 32 Alle registrierten Objekte durch QTP und die erzeugten Schritte im Keyword View

Im nächsten Schritt soll die Zoomfunktion getestet und aufgezeichnet werden. HP QTP ist allerdings nicht in der Lage den „Plus“ Button innerhalb des Kartenausschnittes zu

erkennen. Egal wie oft auf diesen Button drückt wird, HP QTP ist dieses Objekt völlig ungekannt und kann es nicht im Object Repository mit seinen Eigenschaften speichern. Demzufolge wird auch kein neuer Schritt im Testscript erzeugt.

Alternative (fehlerbehaftete) Möglichkeiten

HP QTP stellt für das im letzten Abschnitt gezeigte Problem zwei Möglichkeiten zur Verfügung. Zum einen kann der Test fortgesetzt werden, indem während der Aufnahme der Low-Level-Aufnahmemodus eingestellt wird. Diese Methode ermöglicht das Klicken auf den „Plus“ Button, damit die gewünschte Funktion ausgeführt werden kann.

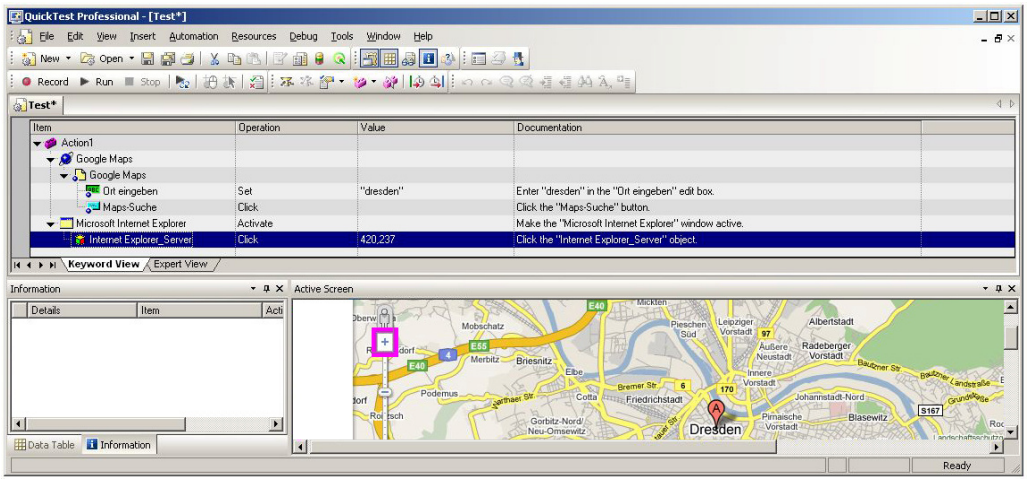

Abbildung 33 Aufgezeichneter Schritt durch den Low-Level-Aufnahmemodus im Keyword View

Zum anderen kann der Bereich, in dem sich der „Zoom“ Button befindet, als Virtuelles Objekt definiert werden. Ist das der Fall kann HP QTP die Interaktion registrieren und einen neuen Schritt im Testscript erzeugen.

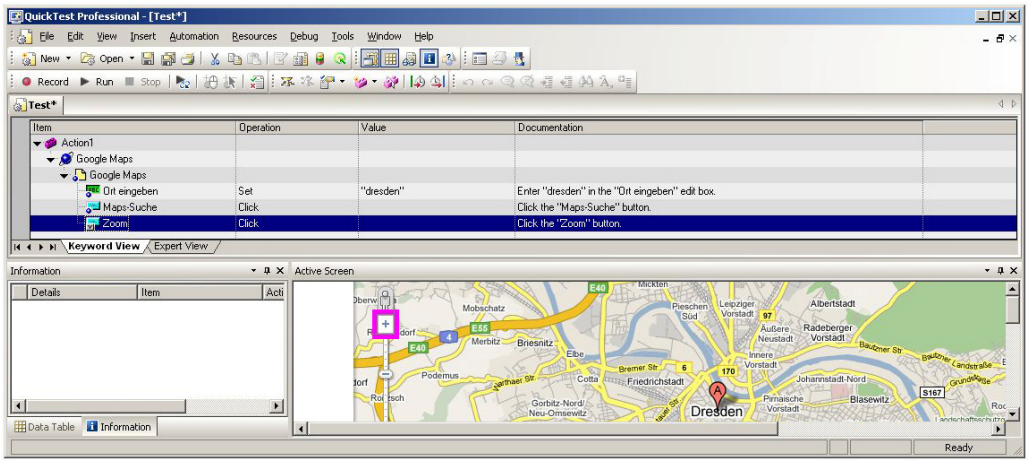

Abbildung 34 Aufgezeichneter Schritt durch einen Virtuellen Button im Keyword View

Diese beiden Methoden besitzen jedoch einen gravierenden Nachteil. Beide sind abhängig von den Koordinaten des Objekts. Da das Objekt durch HP QTP nicht erkannt wurde, wird die Funktion (in diesem Fall das Klicken auf den „Plus“ Button) auf einen bestimmten Bereich bezogen. Wird nach Fertigstellung des Testscriptes ein weiterer Schritt hinzugefügt, der beispielsweise die Detailansicht auf der linken Seite ausblendet, bevor in den Kartenabschnitt gezoomt werden soll, verändert sich die Position des „Plus“ Buttons.

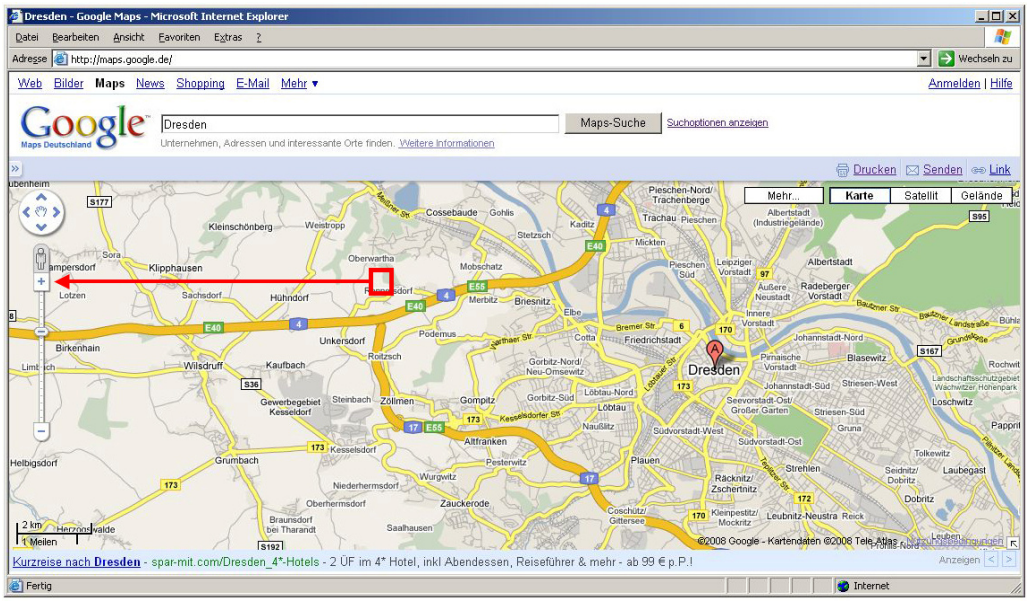

Abbildung 35 Veränderte Position des „Plus“ Buttons im Kartenabschnitt

Keine der beiden Methoden kann jetzt noch den Zoomvorgang durchführen, da sich die Position und demzufolge die Koordinaten des Buttons verändert haben. Bei einer Ausführung des Tests würde HP QTP nicht einmal bemerken, das ein Fehler aufgetreten ist. Nach dem Ausblenden der Detailanzeige führt HP QTP den nächsten Schritt aus und wertet diesen als erfolgreich. Jedoch wird der Klick nicht auf dem Button ausgeführt. Es wird lediglich ein Klick in einen leeren Bereich bzw. in einem Bereich der Karte durchgeführt, da sich der Button an einer anderen Stelle befindet. Die gewünschte Funktion wird dabei allerdings nicht ausgeführt.

Besitzt der Button kein Zoomverhalten, sondern verweist auf eine andere Webseite, so können mögliche darauffolgende Schritte nicht ausgeführt werden. HP QTP nimmt nur an, dass der Button gedrückt wurde. Stattdessen wurde die gewünschte Funktion nicht ausgeführt und ein Seitenwechsel wurde nicht vollzogen. Das im nächsten Schritt gesuchte Objekt kann demzufolge nicht gefunden werden und HP QTP gibt eine

Fehlermeldung zurück. Auslöser des Fehlers war allerdings der vorhergehende Schritt und nicht dieser.

4.4. Ableitung eines Lösungsansatzes

HP QTP nutzt intern in der Regel API-Schnittstellen zur Objekterkennung. Bei verschiedenen Anwendungen kann es allerdings zu Problemen mit der Objekterkennung durch HP QTP kommen, wie es gewissermaßen im angeführten Beispiel mit der Google-Map-Applikation der Fall war.
Deshalb soll als Alternative eine gänzlich andere Herangehensweise bereitgestellt werden, die auf der tatsächlichen grafischen Ausgabe am Bildschirm beruht. Es sollte möglich sein, regelmäßig einen Screenshot der gesamten Anwendung auszulösen und anhand eines vorgegebenen Vergleichsmusters festzustellen, ob und wo ein bestimmtes Element in der Anwendung tatsächlich zu sehen ist, unabhängig von den Aussagen der HP QTP-Objekterkennung.

4.5. Beschreibung der Lösungsstrategie

Da ein Objekt anhand eines Vergleichsmusters gesucht und erkannt werden soll, muss in einem ersten Schritt eine Möglichkeit realisiert werden, die es einem Automatisierer erlaubt, ein Referenzbild vom erwarteten Aussehen des Objekts anzufertigen. Dazu soll ein kleines Tool entwickelt werden, mit dessen Hilfe das Ausschneiden eines Objekts aus dem Screenshot der zu testenden Anwendung möglich ist. Das Werkzeug soll eine grafische Benutzeroberfläche mit einem Anzeigebildschirm und verschiedenen Buttons zur Bedienung zur Verfügung stellen. Das Programm soll nach Aufforderung einen Screenshot der zu testenden Anwendung erstellen und diesen im Anzeigebildschirm wiedergegeben. Der Anwender kann dann einen beliebigen Bereich markieren und diesen Ausschnitt als Referenzbild im Dateisystem speichern. Da einige Elemente in ihrer Darstellung sehr klein sind, sollte das Werkzeug zum Ausschneiden der Objekte eine Zoom-Funktion besitzen, um die Aufnahme je nach Bedürfnis des Anwenders zu vergrößern oder zu verkleinern.
Im zweiten Schritt soll die Funktionalität von HP QTP mit einer Suchfunktion erweitert werden, die auf der tatsächlichen grafischen Ausgabe am Bildschirm beruht und das zuvor als Referenzbild im Dateisystem gespeicherte Steuerelement in der Anwendung sucht und die Koordinaten bei Erfolg oder einen Fehlerwert zurückgibt.

Zur Erweiterung der Funktionalität stehen zwei Möglichkeiten zur Verfügung. Die erste Methode wäre die Erweiterung durch ein Add-in, wie es beispielsweise mit dem Web-Add-in für HP QTP gehandhabt wird, damit Objekte in Webanwendungen erkannt werden können. Diese Methode ist allerdings sehr umständlich und zeitaufwendig. Eine wesentlich schnellere und einfachere Möglichkeit ist die Funktionserweiterung von HP QTP durch eine ActiveX-DLL. Das Kürzel DLL steht dabei für Dynamic Link Library und bezeichnet im allgemeinen eine Dynamische Laufzeitbibliothek oder einfacher gesagt eine dynamisch verlinkbare Bibliothek, die nützliche Funktionen beinhaltet, die zur Verwendung in unterschiedlichen Anwendungen zum Einsatz kommen können. [37] Tritt in einem Programm ein Verweis zu dieser DLL-Datei auf, wird der Programmcode bzw. die Funktion ausgeführt. Die übergebenen Argumente bzw. Variablen können dabei bearbeitet oder verändert und als Rückgabewerte an die Anwendung zurück gegeben werden. Ein klarer Vorteil dieser DLL-Dateien ist, dass diese von mehreren Anwendungen sogar gleichzeitig verwendet werden können. [36] Die entwickelte Funktionalität könnte dem entsprechend nicht nur in HP QTP zum Einsatz kommen, sondern auch in anderen Programmen verwendet werden. Ein weiterer Vorteil ist die einfache Aktualisierung der DLL. Ändern sich Funktionen in der DLL, müssen die Anwendungen, die diese DLL verwenden, nicht neu kompiliert oder neu verknüpft werden, solange sich die Argumente und Rückgabewerte der Funktion nicht ändern. [36] Dem entsprechend soll mittels Visual Basic eine ActiveX-DLL entwickelt werden, die die geforderte Funktionalität zur Verfügung stellt.

5. Vorstellung einer alternativen Objekterkennung

5.1. Tool zur Referenzbilderzeugung

5.1.1. Vorüberlegung

Zum Anfertigen der Referenzbilder muss ein Werkzeug entwickelt werden, welches eine grafische Oberfläche besitzt, auf der eine Picture-Box (Anzeigebildschirm) und sechs Steuerelemente (Buttons) zur Bedienung des Tools platziert werden. Im Anzeigebildschirm wird der durch den Benutzer ausgelöste Screenshot der zu testenden Applikation, aus dem das Referenzobjekt ausgeschnitten werden soll, angezeigt. Danach müsste der Benutzer einen Markierungsrahmen mittels der Maus zeichnen können, um den Bereich zu markieren, der später als Referenzobjekt gespeichert werden soll. Da einige Objekte in ihrer Darstellung sehr klein sind, ist eine Zoom-Funktion von Vorteil, die den Screenshot vergrößert darstellt, damit eine exakte Markierung gewährleistet werden kann. Des Weiteren ist eine Funktion zum Löschen des Anzeigebildschirms wünschenswert, sowie ein Button zum Beenden des Tools. Das Entwickelte Tool erhält im Anschluss der Programmierung den Namen „CaptureIt“.

5.1.2. Praktische Umsetzung

Die Entwicklung des CaptureIt-Tools wird mittels VB 8 als Teil von Visual Studio (VS) 2005 umgesetzt, mit der einfache Windows-Benutzeroberflächen erstellt werden können. Nach dem Start von VS 2005 muss ein neues Projekt angelegt werden. Dazu wird aus der Liste der vorinstallierten Vorlagen für VB die Windows-Anwendung ausgewählt, ein Projektname vergeben und der Speicherort festgelegt. VS erzeugt nach der Bestätigung automatisch eine neue Projektmappe in der eine neue Form (Windows-Benutzeroberfläche) vorhanden ist, die je nach den Bedürfnissen des Benutzers verändert werden kann.

Im ersten Schritt wird die Benutzeroberfläche entsprechend der Vorüberlegung angepasst. Dazu werden sechs Steuerelemente auf der linken Seite der Form platziert und der Reihe nach beschriftet. Der erste Button bekommt die Aufschrift „Screenshot“, die beiden darauf folgenden werden mit „Zoom (+)“ und „Zoom (-)“ gekennzeichnet, die nächsten mit „Save“ sowie „Clear“ und der letzte mit „Exit“. Zur späteren Anzeige

des Screenshots der zu testenden Anwendung wird noch eine PictureBox benötigt. Da allerdings der Screenshot der Anwendung sehr groß sein wird und in der PictureBox nicht komplett zur Anzeige gebracht werden kann, benötigt man eine Scroll-Funktion. Die PictureBox selber bietet diese Möglichkeit nicht an. Deshalb wird in diesem Fall ein Panel verwendet, mit der die Grafik bzw. die PictureBox verschoben werden kann. Es wird also zuerst ein Panel auf der Form platziert, mit dem der Scroll-Bereich dargestellt wird, dessen *AutoScroll* Eigenschaft auf *True* gesetzt wird. Erst danach wird die PictureBox (*SizeMode=AutoSize*) auf dem Panel platziert.

Im Anschluss an die Gestaltung der Benutzeroberfläche erfolgt die Vorbereitung für das Auslösen des Screenshots. Dazu werden im oberen Teil des Skriptes die drei API-Funktionen (Erläuterung des Begriffs API-Funktion im Abschnitt „Notwendige Funktionen“) `BitBlt`, `GetDC` und `ReleaseDC` deklariert. Die Funktion `BitBlt` ermöglicht später das kopieren eines Anzeige-Gerätekontextes (engl.: Device-Context , Abk.: DC) in einen anderen DC. [51] Der Aufbau der Funktion sieht folgender Maßen aus:

```
<DllImport("gdi32.dll")> _
Private Shared Function BitBlt( _
   ByVal hDestDC As IntPtr, _
   ByVal x As Integer, _
   ByVal y As Integer, _
   ByVal nWidth As Integer, _
   ByVal nHeight As Integer, _
   ByVal hSrcDC As IntPtr, _
   ByVal xSrc As Integer, _
   ByVal ySrc As Integer, _
   ByVal dwRop As Integer) As Integer
End Function
```

Listing 7 BitBlt-Funktion

`hDestDC` erwartet den DC des Zielobjekts, in welches später die Bilddaten des Quellobjekts kopiert (geblittet [52]) werden. `x` entspricht dabei der horizontalen Koordinate und `y` der vertikalen Koordinate zu dem das Zielobjekt geblittet werden soll. Mit den beiden Parametern `nWidth` und `nHeight` wird die neue Breite bzw. Höhe des Zielobjekts festgelegt. Darüber hinaus erwartet `hSrcDC` den DC des Quellobjekts, dessen Inhalt in das Zielobjekt geblittet werden soll und `xSrc` sowie `ySrc` die horizontale bzw. vertikale Koordinate von der Position, an welcher die Grafik kopiert werden soll. Nachfolgend wird eine `dwRop`-Konstante beigefügt. In diesem Fall wird die `SRCCOPY`-Konstante verwendet, die das Kopieren des Inhalts der Quelle ins Ziel

veranlasst. [51] Des Weiteren wird die `GetDC`-Funktion benötigt, die das Handle eines DC für den Clientbereich eines bestimmten Fensters oder den gesamten Bildschirm (Desktop) ermittelt. [53] Wird dieser DC nicht mehr gebraucht, kann er mittels `ReleaseDC` wieder freigegeben werden, so dass andere Anwendungen diesen wieder verwenden können. [54]

VB Programme können so geschrieben sein, dass diese streng Befehl auf Befehl abarbeiten. In der Praxis sieht es jedoch anders aus. Gerade bei Windows-Anwendungen sind die erstellten Programme Ereignisorientiert, d.h., der Programmcode wird ausgeführt wenn etwas passiert. Das Ereignis (Event) kann dabei durch den Benutzer ausgelöst werden, aber auch eine Folge von anderen Befehlen sein. [55] Da der Screenshot der zu testenden Anwendung ausgelöst werden soll, wenn ein Benutzer des Werkzeuges den Screenshot-Button drückt, muss der Programmcode zwischen `Private Sub Screenshot_Click(…) Handles Screenshot.Click` und `End Sub` geschrieben sein. Als erstes wird im Screenshot-Funktionsablauf der `ZoomFaktor` auf `1` gesetzt. Die eins soll dabei für den normalen Zustand des Screenshots stehen. Die Grafik wurde also noch nicht gezoomt. Da ein Desktopscreenshot ausgelöst werden soll muss als nächstes der Befehl `Me.Visible=False` im Programmcode stehen, damit die Benutzeroberfläche des Werkzeugs während der Screenshotaufnahme ausgeblendet wird. Erst nach Abarbeitung der Skriptbefehle zum Screenshot soll die Benutzeroberfläche durch den Befehl `Me.Visible=True` wieder eingeblendet werden. Da jedoch die Abarbeitung der Schritte sehr schnell ausgeführt wird muss dem Programm etwas Rechenzeit zur Verfügung gestellt werden, damit die Benutzeroberfläche lang genug ausgeblendet wird und im Screenshot nicht zu sehen ist. Durch den Aufruf `Call Wait(500)` wird die Funktion `Wait`, die die geforderte Rechenzeit zur Verfügung stellt, aufgerufen und abgearbeitet.

```
Private Sub Wait(ByVal Milliseconds As Integer)
  Dim time As Date
  time = Now.AddMilliseconds(Milliseconds)
  Do While time > Now
    Application.DoEvents()
  Loop
End Sub
```

Listing 8 Wait-Funktion

Dabei wird die aktuelle Zeit genommen und mit 500 Millisekunden, die mit dem Funktionsaufruf übergeben werden, addiert. Danach wird eine Schleife durchlaufen, bis der durch die Addition errechnetet Zeitwert erreicht ist. Im Anschluss wird der Programmcode nach dem Funktionsaufruf und damit der Screenshotvorgang weiter ausgeführt. Dazu wird im nächsten Schritt ein neues Bitmap-Image mit den Abmaßen des Desktops und einer Farbtiefe von 24-Bit erzeugt. Anschließend wird das Image der Variable `g1` vom Typ `Graphics` übergeben und ein DC für den Desktop-Screen und die Bitmap erzeugt.

```
'Bitmap mit den Maßen des Desktop erzeugen
img = New Bitmap(Screen.PrimaryScreen.WorkingArea.Width, _
                 Screen.PrimaryScreen.WorkingArea.Height, _
                 Imaging.PixelFormat.Format24bppRgb)
g1 = Graphics.FromImage(img)
'DC für den Desktop-Screen erzeugen
dc1 = GetDC(0)
'DC für die Bitmap erzeugen
dc2 = g1.GetHdc()
```

Listing 9 Neues Bitmap-Image und die DCs erzeugen

Damit sind alle Schritte für den Screenshot abgeschlossen und die Bilddaten können vom Desktop-Screen mittels der `BitBlt`-Funktion in die Bitmap kopiert werden.

```
BitBlt(dc2, 0, 0, Screen.PrimaryScreen.WorkingArea.Width, _
      Screen.PrimaryScreen.WorkingArea.Height, dc1, 0, 0, SRCCOPY)
```

Listing 10 Anwendung der BitBlt-Funktion

Das Zielobjekt ist das neu erstellte Bitmap-Image, welches in diesem Fall die Bilddaten des Desktop-Screens erhält. Damit kann nach der Freigabe des Screen-DCs und Bitmap-DCs, durch die `ReleaseDC`-Funktion, das Bitmap-Image der PictureBox übergeben und zur Anzeige gebracht werden.

```
'Screen-DC und Bitmap-DC freigeben
ReleaseDC(0, dc1)
g1.ReleaseHdc(dc1)
'Zuweisen der Grafik an die PictureBox
PictureBox1.Image = img
```

Listing 11 Freigabe der DCs und Zuweisung der Grafik an die PictureBox

Nachdem der Desktop-Screenshot in der PictureBox sichtbar ist, treten die Funktionen in den Vordergrund, die die Mausaktionen des Benutzers in der PictureBox auswerten, damit ein markierter Bereich im Screenshot als Referenzbild im Dateisystem gespeichert werden kann. Dazu werden drei Funktionsabläufe benötigt, bei denen die Ereignisse:

- Drücken der linken Maustaste,
- Zeichnen des Markierungsrahmens (Mausbewegung bei gedrückter linker Maustaste) und
- Loslassen der linken Maustaste

ausgewertet und verschiedene Aktionen durchgeführt werden. Beim `MouseDown`-Event (Maustaste gedrückt) für die linke Maustaste wird zuerst die Grafik in der PictureBox durch einen `Refresh`-Aufruf neu gezeichnet, damit mögliche Markierungsrahmen die vom Benutzer gezogen wurden gelöscht werden. Danach werden die Koordinaten der Punkte `p1` und `p2`, an denen der Benutzer mit der linken Maustaste in die Grafik klickt, gespeichert. Diese Koordinaten werden später zur Darstellung des Markierungsrahmens und zur Größenberechnung bzw. zur Lokalisierung des Referenzbildes benötigt.
Mit der Abfrage, ob die linke Maustaste gedrückt ist, beginnt die `MouseMove`-Funktion (Mausbewegung). Ist das der Fall wird fortlaufend der Programmcode dieser Funktion ausgeführt. Dabei wird immer wieder der Markierungsrahmen gelöscht, die neue Position von `p2` gespeichert und der Markierungsrahmen neu gezeichnet.

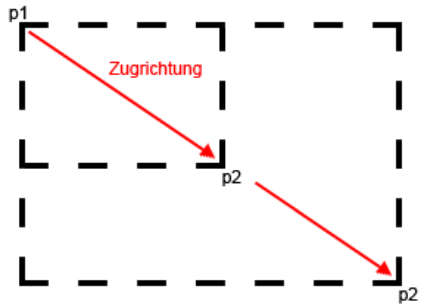

Abbildung 36 Markierungsrahmen zeichnen

Dieser Funktionsablauf wird solange ausgeführt, bis die linke Maustaste los gelassen wird. Danach bleibt der Markierungsrahmen, der um das gewünschte Objekt gezogen wurde, stehen und der Programmcode des `MouseUp`-Events (Maustaste losgelassen) wird ausgeführt.

Diese `MouseUp`-Funktion soll die Größe des Rechtecks innerhalb des erzeugten Markierungsrahmens berechnen, die gleichzeitig die Größe des Referenzbildes widerspiegelt und später beim Speichervorgang benötigt wird. Die Zugrichtung des Rahmens muss dabei beachtet werden.

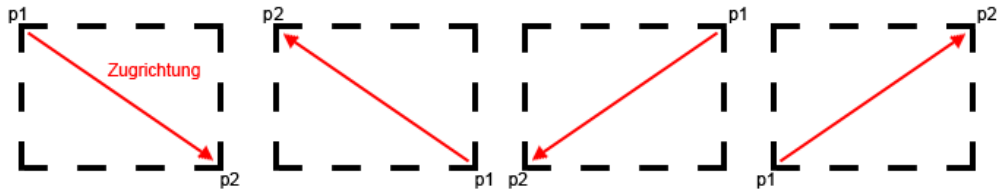

Abbildung 37 Zugrichtung vom Markierungsrahmen

Wird beispielsweise der Rahmen vom linken oberen Bereich der PictureBox in den rechten unteren Bereich gezogen, dann sind die x- und y-Koordinaten von Punkt `p2` größer als die von Punkt `p1` und die Berechnung der Rahmenbreite lautet `DRx=(p2.X-p1.X)-2`. `DRx` steht dabei für die Rechteckbreite und der Wert `2` wird vom Ergebnis abgezogen, da die Rahmenstärke auf beiden Seiten jeweils einem Pixel entspricht und zum Referenzbild nicht dazu gehört. Wird der Rahmen allerdings in umgekehrter Richtung aufgezogen, sind die Koordinaten des ersten Punktes größer und die Rechnung muss neu gestaltet werden, damit kein negativer Wert als Ergebnis stehen bleibt. Die Prüfung des Ergebnisses wird mittels einer Abfrage realisiert. Sollte das erste Ergebnis einen negativen Wert besitzen werden die beiden Koordinatenwerte vertauscht und die Berechnung wird erneut ausgeführt. Diese Berechnung erfolgt gleichermaßen für die Breite und die Höhe des Rechtecks.

```
DRx = (p2.X - p1.X) - 2
If DRx < 0 Then
   DRx = (p1.X - p2.X) - 2
End If
```

Listing 12 Berechnung der Rechteckbreite

Wurde der Markierungsrahmen um das gewünschte Objekt gezogen, kann die Grafik gespeichert werden. Dazu wird die `Save_Click`-Funktion ausgeführt, die abgearbeitet wird, wenn ein Benutzer den „Save“-Button drückt.

Zum Speichern der Grafik wird zuerst ein Rechteck erstellt, das die Daten des Referenzbildes enthält. Dazu gehören die Koordinaten, an denen sich die Grafik im Screenshot befindet und die Größe der Grafik, die ausgeschnitten und als

Referenzobjekt gespeichert werden soll. Dabei ist immer die Zugrichtung des Markierungsrahmens zu beachten.

```
ClipRectangle = New Rectangle(CInt(p1.X / ZoomFaktor) + 1, _
                              CInt(p1.Y / ZoomFaktor) + 1, _
                              CInt(DRx / ZoomFaktor), _
                              CInt(DRy / ZoomFaktor))
```

Listing 13 Rechteck für Referenzbilddaten erstellen

Der im Listing 25 gezeigte Programmcode bezieht sich auf das Rechteck, dass mit einem Markierungsrahmen vom oberen linken in den unteren rechten Bereich der PictureBox gezogen wurde. Nach dem Aufruf `New Rectangle` (neues Rechteck) folgen die Parameter zur Beschreibung des Rechtecks. Der erste Parameter bestimmt die x-Koordinate der linken oberen Ecke an der sich das Rechteck in der PictureBox befindet und der zweite Parameter die y-Koordinate. Die beiden folgenden Parameter bestimmen die Breite und die Höhe des Rechtecks. Alle Variablen müssen durch den `ZoomFaktor` geteilt werden. Dieser ist zum jetzigen Zeitpunkt `1`, da der Screenshot in der PictureBox noch nicht gezoomt wurde. Als nächstes wird ein neues Bitmap-Image angelegt, das die Breite und Höhe des gerade erzeugten Rechtecks und das Pixelformat des Screenshots erhält. Diese Bitmap wird dem Grafikobjekt `g2` übergeben und das Referenzbild wird mit dem Befehl `g2.DrawImage` gezeichnet.

```
'Bitmap mit den Maßen des Rechtecks erzeugen
ref = New Bitmap(ClipRectangle.Width, ClipRectangle.Height, _
                 img.PixelFormat)
'Bitmap der Variablen g2 übergeben
g2 = Graphics.FromImage(ref)
'Referenzbild zeichnen
g2.DrawImage(img, 0, 0, ClipRectangle, GraphicsUnit.Pixel)
```

Listing 14 Referenzbild zeichnen

`img` gibt das Quellobjekt und `ClipRectangle` den Teil des Objektes an, der im neuen Bitmap-Image gezeichnet werden soll. Damit liegt das Referenzbild in der Variablen `ref` vor und kann vom Benutzer im Dateisystem gespeichert werden. Dazu wird die sogenannte SaveFileDialog-Klasse verwendet, die dem Benutzer die windowstypische Benutzeroberfläche zur Auswahl des Speicherortes und zur Vergabe des Dateinamens angezeigt wird.

```
'SaveFileDialog
Dim SaveFile As New SaveFileDialog
'Parameter für den SaveFileDialog setzen
With SaveFile
   .Title = "Save Bitmap"
   .FileName = ""
   .DefaultExt = "bmp"
   .Filter = "Bitmap (*.bmp)|*.bmp"
   .OverwritePrompt = True
End With
'OK-Button im SaveFileDialog gedrückt, dann Anweisung ausführen
If SaveFile.ShowDialog() = Windows.Forms.DialogResult.OK Then
      ref.Save(SaveFile.FileName, _
         System.Drawing.Imaging.ImageFormat.Bmp)
End If
```

Listing 15 SaveFileDialog-Klasse

Die Parameter für den SaveFileDialog müssen dabei individuell gesetzt werden. Der Titel im SaveFileDialog wird im Parameter `.Title` festgelegt. Der folgende Parameter gibt den Dateinamen an, der im SaveFileDialog angezeigt werden soll. Da der Dateiname vom Benutzer selbst vergeben werden soll, ist dieser Parameter leer. Die zum Speichern verfügbaren Dateierweiterungen (`.DefaultExt`) und Dateitypen (`.Filter`) werden ebenfalls festgelegt. Zur Auswahl soll hier nur das Bitmap-Format stehen. Der letzte Parameter `.OverwritePrompt` überprüft ob der Dateiname am angegebenen Speicherort vorhanden ist und öffnet bei einer Übereinstimmung ein kleines Fenster in dem des Überschreiben der Datei bestätigt werden muss. Danach wird das Bitmap-Image bzw. das Referenzbild durch einen Klick auf den OK-Button, mit dem durch den Benutzer vergebenen Dateinamen, im Dateisystem gespeichert.

Da einige Referenzobjekte in ihrer Darstellung sehr klein sind, besitzt dieses Tool eine Zoom-Funktion, damit diese exakter zugeschnitten werden können. Warum das exakte Zuschneiden der Referenzbilder so wichtig ist verdeutlicht die folgende Abbildung:

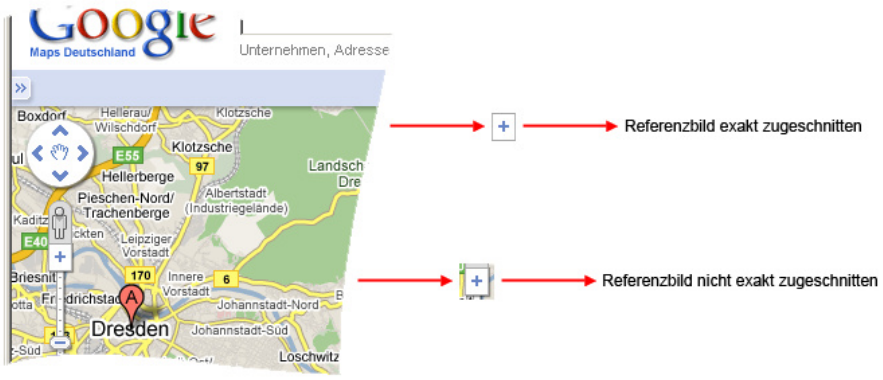

Abbildung 38 Exaktes Zuschneiden der Referenzbilder

Da sich das Kartenmaterial bei einem Testlauf ständig ändern wird, muss der „Plus“-Button korrekt ausgeschnitten und als Referenz abgespeichert werden. Wird dabei zu viel von der im Hintergrund angezeigten Karte mit gespeichert, könnte die Suchfunktion nach diesem Objekt fehl schlagen. Dem zufolge kann der Benutzer stufenweise den Screenshot in der PictureBox vergrößern oder verkleinern und somit das Referenzobjekt gründlicher zuschneiden. Dabei stehen zwei Stufen zur Verfügung. In der ersten Stufe besitzt die gezoomte Grafik die doppelte Größe im Vergleich zum Original und in der zweiten Stufe die vierfachen Abmessung. Abhängig vom dazugehörigen `ZoomFaktor` wird dann der Programmcode im `ZoomIn_Click`-Event („Zoom(+)“-Button gedrückt) abgearbeitet.

```
'Neues Bitmap mit den doppelten Maßen des Originals erzeugen
gs1 = PictureBox1.CreateGraphics()
img2 = New Bitmap(img.Width * 2, img.Height * 2, _
                  Imaging.PixelFormat.Format24bppRgb)
gs1 = Graphics.FromImage(img2)
'Neues Bitmap zeichnen
gs1.DrawImage(img, _
              New Rectangle(0, 0, img.Width * 2, img.Height * 2), _
              New Rectangle(0, 0, img.Width, img.Height), _
              GraphicsUnit.Pixel)
```

Listing 16 ZoomIn-Event für die erste Stufe beim Zoomen (doppelte Größe zum Original)

Dabei wird zuerst ein neues Bitmap-Image erzeugt, dass die doppelten bzw. vierfachen Maße des Originals erhält. Danach kann das neue Bitmap-Image mittels `.DrawImage` gezeichnet werden, wobei die Bilddaten für den Zeichenvorgang aus dem Quellobjekt herangezogen werden. Im folgenden Schritt wird das neue Bitmap-Image der PictureBox übergeben und der `ZoomFaktor` wird erhöht. Dem zufolge erhält der Benutzer einen sichtbaren Zoom-Effekt im Anzeigebildschirm und kann dadurch das Referenzobjekt genauer ausschneiden.

Sollte die gezoomte Grafik für den Benutzer zu groß sein, kann diese jederzeit mit dem „Zoom(-)“-Button verkleinert werden. Dazu wird lediglich im Programmcode der PictureBox das zum jeweiligen `ZoomFaktor` entsprechende Image übergeben. Des weiteren kann die PictureBox mit dem „Clear“-Button gelöscht, oder das gesamte Programm mit dem „Exit“-Button geschlossen werden. Dabei werden die Funktionsabläufe in den dazugehörigen Events abgearbeitet.

5.1.3. Anwendung

Damit ein Referenzbild für die Suchfunktion erstellt werden kann, sollte zuerst vom Anwender die Applikation geöffnet werden, in der später das Objekt gesucht werden soll. Wechselt der Benutzer wieder zum CaptureIt-Tool und drückt den „Screenshot“ Button wird ein Desktop-Screenshot ausgelöst, auf dem ebenfalls die zu testende Applikation abgebildet ist. Die entstandene Grafik wird anschließend in der Picture-Box des Tools für die weitere Bearbeitung zur Anzeige gebracht. Ist das Element, das ausgeschnitten werden soll sehr klein, kann der Benutzer mit Hilfe der Zoom-Funktion den Screenshot vergrößern. Durch einen Klick auf den „Zoom (+)“ Button wird die Größe des Bildes in der Anzeige verdoppelt und bei einem wiederholten klick sogar vervierfacht. Dadurch wird der Screenshot vergrößert dargestellt und eine genaue Markierung einzelner Elemente kann exakter durchgeführt werden.
Zur Kennzeichnung eines bestimmten Bereichs muss der Benutzer mit der Maus in den Anzeigebildschirm navigieren, die linke Maustaste drücken und kann durch das ziehen der Maus einen Markierungsrahmen aufziehen. Befindet sich das gewünschte Objekt in diesem Rahmen kann das Bild, durch einen Klick auf den „Save“ Button, gespeichert werden. Ist dies nicht der Fall kann der Markierungsrahmen neu platziert werden.

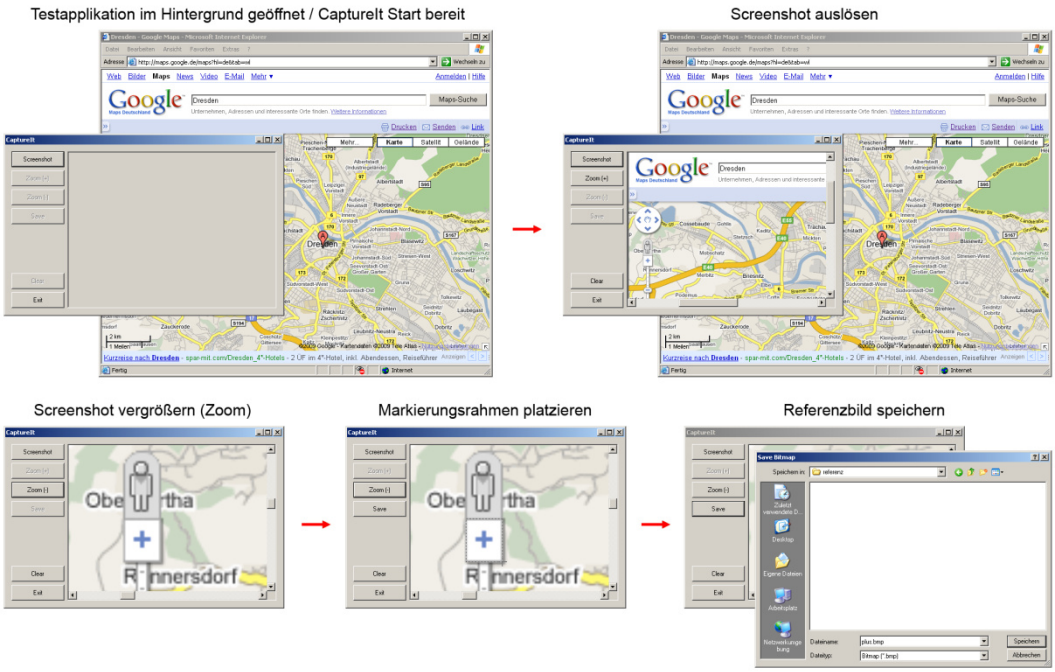

Abbildung 39 Funktionsablauf zur Referenzbilderzeugung

Sollte eine Aufnahme der Testapplikation nicht gelungen sein kann der Screenshot-Vorgang jederzeit wiederholt werden. Dabei bleibt es dem Benutzer überlassen, ob er

vorher durch „Clear“ den Anzeigebildschirm löscht oder den Vorgang sofort durchführt. Mit „Exit“ wird das Tools geschlossen.

5.2. ActiveX-DLL zur Funktionserweiterung

5.2.1. Theoretischer Ansatz

Das Bitmap-Format ist ein Dateiformat zur verlustfreien Speicherung von Rastergrafiken und wird unter Windows mit der Dateiendung *.bmp gekennzeichnet. Ein deutlicher Vorteil des Bitmap-Formats ist sein relativ einfacher Aufbau. Darüber hinaus ist es das Standardformat unter Windows und wird zum Speichern von geräte- und anwendungsunabhängigen Grafiken verwendet. Der Nachteil dieses Formats ist der Speicherbedarf im Dateisystem, der im Vergleich zu den Formaten *.jpg oder *.png wesentlich höher ist.

Bitmaps bestehen aus drei Teilen, dem Dateikopf (BITMAPFILEHEADER) in dem Daten zur Identifikation gespeichert sind, dem Informationsblock (BITMAPINFOHEADER) der wertvolle Informationen über die Eigenschaften der Bitmap enthält und einem Bereich in dem die eigentlichen Bilddaten hinterlegt sind.

Dateikopf (BITMAPFILEHEADER)
Informationsblock (BITMAPINFOHEADER)
Bilddaten

BITMAPFILEHEADER (Größe: 14 Byte)

Name	Größe	Inhalt
bfType	4 Byte	Dateiformat (BM = Bitmap)
bfSize	4 Byte	Dateigröße
bfReserved	4 Byte	NULL
bfOffBits	4 Byte	Position der Bilddaten

BITMAPINFOHEADER (Größe: 40 Byte)

Name	Größe	Inhalt
biSize	4 Byte	Größe der BITMAPINFOHEADER-Struktur
biWidth	4 Byte	Breite der Bitmap in Pixeln
biHeight	4 Byte	Höhe der Bitmap in Pixeln
biPlanes	2 Byte	Farbebenen
biBitCount	2 Byte	Farbtiefe der Bitmap
biCompression	4 Byte	Komprimierung
biSizeImage	4 Byte	Größe der Bilddaten
biXPelsPerMeter	4 Byte	Horizontale Auflösung
biYPelsPerMeter	4 Byte	Vertikale Auflösung
biClrUsed	4 Byte	Anzahl der Einträge der Farbtabelle
biClrImportant	4 Byte	Anzahl sämtlicher im Bild verwendeten Farben

Abbildung 40 Aufbau einer Bitmap-Datei

Die digitalisierten Bilddaten liegen im Speicher in einer Art Raster, das durch horizontale und vertikale Linien mit jeweils gleichem Abstand in Rechtecke aufgeteilt ist. Man könnte also sagen, dass die Bilddaten in mehrere Bildzeilen aufgeteilt sind, in denen sich die einzelnen Pixel hintereinander aufreihen. Die Bild-Pixel werden in der

Regel mit einer Farbtiefe von 1, 4, 8, 16, 24, 32, 48, oder 64 Bit pro Pixel gespeichert. Die Anzahl der Bits die einem einzelnen Pixel zugeordnet werden, bestimmt die Anzahl der Farben die zur Darstellung des Pixels verwendet werden können. Wird beispielsweise ein Pixel durch 24 Bit repräsentiert, dann kann dieser bestimmte Pixel durch eine von Rund 16,7 Millionen verschiedenen Farben (2^{24} = 16777216) abgebildet werden. [38, 39, 40, 41]

Im Fokus der später durchgeführten Imagesuche stehen Bitmaps mit einer Farbtiefe von 24-Bit. Diese besitzen den Vorteil, dass zur Darstellung eines farbigen Pixels der 24-Bit-Farbwert in drei Farbanteile zerfällt. Jeder Farbkanal benötigt ein Byte Speicherplatz (1 Byte = 8 Bit → 3 • 8 Bit = 24 Bit) und repräsentiert eine der drei Grund- oder Primärfarben Rot (R), Grün (G) und Blau (B). Je nach Mischungsverhältnis entstehen dann die verschieden Farben. Die Farbanteile werden in 256 Stufen (von 0 bis 255) angegeben. Sind die Anteile jeder Farbe null, entsteht Schwarz, mit den Maxima von 255 entsteht Weiß. Dem zufolge lässt sich also ein 24-Bit-Farbwert als eine RGB-Zahl darstellen.

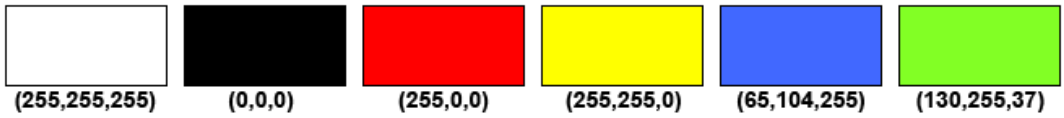

Abbildung 41 Einige Farben und die dazugehörigen RGB-Werte

In einer 24-Bit-Bitmap liegen also jeweils immer 3 Byte nebeneinander, die nach dem oben genannten Schema je ein Farbpixel bilden. Den visuellen Aufbau einer 4x2 Pixel großen Grafik könnte man sich im Speicher also wie folgt vorstellen:

	Pixel 1	Pixel 2	Pixel 3	Pixel 4
Zeile 1	R G B	R G B	R G B	R G B
Zeile 2	R G B	R G B	R G B	R G B

Abbildung 42 Visueller Aufbau einer 4x2 Pixel großen Grafik

Durch diese eindeutige Positionierung der einzelnen Pixel und die Zuordnung des dazugehörigen Farbwertes, beziehungsweise der drei Farbanteile, lassen sich Bitmaps eindeutig beschreiben. [42, 43]

Mit diesem Faktenwissen wird eine Funktion aufgebaut, die es ermöglicht ein Referenzbild im Screenshot einer Testapplikation zu suchen. Um dies zu realisieren,

werden bei beiden Grafiken die einzelnen Pixel, genauer gesagt die Farbwerte der Pixel, miteinander verglichen. Als erstes wird die Farbinformation eines definierten Pixels in der kleinen Grafik (Referenzbild) ermittelt, beispielsweise der RGB-Wert des linken unteren Pixels. Danach muss genau dieser Farbwert in der großen Grafik (Screenshot der Testapplikation) gesucht werden, da es sich dabei um den linken unteren Rand der gesuchten Grafik handeln könnte. Zeile für Zeile vergleicht die Funktion nacheinander die Farbinformationen der Pixel mit dem gegeben RGB-Wert, bis eine Übereinstimmung gefunden wird. Tritt eine Gleichheit der Farbwerte auf, könnte das erste gemeinsame Pixel gefunden worden sein.
Im nächsten Schritt fragt die Funktion nicht nur die Farbinformation des folgenden Pixels im Screenshot ab, sondern auch die Farbinformation des nächsten Pixels im Referenzbild. Danach findet ein wiederholter Vergleich statt. Bei erneuter Übereinstimmung der Farbwerte ist dieser Schritt zu wiederholen, bis der letzte Pixel vom Referenzbild mit dem des Screenshots auf Gleichheit überprüft worden ist. Sollte sich vorher eine Abweichung der Farbwerte ergeben, muss wieder die Farbinformation des ersten definierten Pixels im Referenzbild ermittelt werden und die Suchfunktion am Folgepixel der ersten Übereinstimmung fortgesetzt werden.
Bei einem durchgehend positiven Wertevergleich der Farbinformation, vom ersten bis zum letzten überprüften Pixel, war der Funktionsdurchlauf erfolgreich und das gesuchte Objekt wurde gefunden. Danach werden die Koordinaten des absoluten Schnittpunktes ermittelt, die später an HP QTP übergeben werden können.
Da es in einigen Fällen zu kleineren Abweichungen, durch Unschärfe oder ähnliche Grafikfehler, bei der Darstellung einzelner Elemente innerhalb der Testapplikation kommen kann, ist eine geringe Abweichung beim Vergleich der Farbwerte zulässig. Dem Anwender wird dabei die Entscheidung überlassen, welchen Wert die Abweichung maximal betragen darf.

5.2.2. Notwendige Funktionen

API-Funktionen

Zur Realisierung der Suchfunktion wurde die objektorientierte Programmiersprache Visual Basic (VB) verwendet. VB stellt einen recht umfangreichen Befehlsvorrat zur Programmierung von Softwaresystemen zur Verfügung. Dennoch kommt es vor, das man schnell an die Grenzen, mit der zur Auswahl stehenden Funktionalitäten, bei der

Lösung einer Aufgabe stößt. Da man jederzeit auf die zahlreichen Funktionen des Betriebssystems zurückgreifen kann, sind diese Probleme schnell lösbar. Eine API-Funktion, Application Programming Interface, stellt dabei die Schnittstelle zwischen einer Softwareanwendung und dem Betriebssystem dar. Es dient also zur Kommunikation zwischen den beiden System und ermöglicht einen Zugriff auf die Funktionen des Betriebssystems. [44]

SafeArray

Unter Visual Basic gibt es viele Möglichkeiten Bitmaps pixelgenau zu manipulieren. Die ersten beiden Methoden sind `PSet` und `Point`. Mit `Point` kann ein Farbwert eines Pixels von einer angegebenen Position ausgelesen und falls erforderlich mittels `PSet` geändert werden. Zwei weitere Methoden, die im Prinzip dieselbe Funktion erfüllen, heißen `GetPixel` zur Abfrage der Farbinformation und `SetPixel` zur Änderung der Farbinformation. Diese Techniken sind allerdings sehr langsam und für einen schnellen Bildvergleich ungeeignet. VB bietet hierfür eine weitaus bessere Möglichkeit, die das Einlesen von Bitmaps in ein Array innerhalb von Mikrosekunden bewerkstelligt. Befindet sich eine Grafik in einem Array lassen sich mit dieser Technik komplexe Operationen auf die gespeicherte Bitmap anwenden.
Damit die hohe Dynamik von Arrays verwaltet werden kann, wird ein sogenannter Deskriptor benötigt. Beispielsweise wird das dynamische Verschieben von Feldobergrenzen und Felduntergrenzen zur Laufzeit erst durch diesen besagten Deskriptor ermöglicht. Ein Deskriptor kann man sich als einen Speicherbereich mit vordefiniertem Aufbau vorstellen, der sich als Kombination zweier Strukturen darstellen lässt. Der erste Teil der Struktur beschreibt die Arraygrenzen und ist folgender Maßen definiert:

```
Private Type SAFEARRAYBOUND
   cElements As Long
   lLbound As Long
End Type
```

Listing 17 Der erste Teil des SaveArrays (SAFEARRAYBOUND)

Die Gesamtzahl der im Feld enthaltenen Elemente wird dabei durch `cElements` repräsentiert und die Untergrenze, die dem Index des ersten Feldelementes entspricht, durch `lLbound`. `SAFEARRAYBOUND` ist, neben anderen Elementen ein Member der zweiten Struktur, deren Aufbau in folgender Form darzustellen ist:

```
Private Type SAFEARRAY2D
   cDims As Integer
   fFeatures As Integer
   cbElements As Long
   cLocks As Long
   pvData As Long
   Bounds(0 To 1) As SAFEARRAYBOUND
End Type
```

Listing 18 Der zweite Teil des SaveArrays (SAFEARRAY2D)

Das Element `cDims` kennzeichnet dabei die Dimension des Arrays. `fFeatures` gibt Auskunft über den verwendeten Datentyp, `cbElements` wieviel Speicherplatz ein einzelnes Element des Feldes benötigt und `cLocks` gibt den aktuellen Wert des Sperrzähler eines Arrays an. `pvData` beinhaltet einen 4 Byte langer Zeiger, der die eigentlichen Daten referenziert. Diese sind losgelöst vom Deskriptor und befinden sich an beliebiger Stelle im Speicher. Das Element `Bounds(0 To 1)`ist eine Struktur vom Typ `SAFEARRAYBOUND`, welche wie bereits oben erwähnt, die Größe und die Untergrenze des Arrays aufnehmen, wobei der Parameter `(0 To 1)`ein zweidimensionales Array definiert. Dem entsprechend erhält die Deskriptor-Struktur auch die Bezeichnung `SAFEARRAY2D`. [45, 48, 49]

Pointer

Variablen besitzen neben ihrem eigentlichen Wert auch eine Speicheradresse, an der dieser Wert abgelegt wird. Ein Pointer (Zeiger) ist also nichts anderes als die Adresse einer Variable im Arbeitsspeicher.

Eine ganze Reihe von API-Funktionen erfordern die Übergabe dieser Pointer anstelle des Wertes einer Variablen. Die direkte Ermittlung eines Pointer einer Variablen erhält man über die wenig dokumentierten Funktionen:

- `StrPtr` (ermittelt den Pointer eines Strings),
- `ObjPtr` (ermittelt den Pointer eines Objekts),
- `VarPtr` (ermittelt den Pointer einer numerischen Variablen) und
- `VarPtrArray` (ermittelt den Pointer eines Feldes).

Da `VarPtrArray` nicht wie die anderen drei Pointerfunktionen standardmäßig implementiert ist, muss eine eigene Deklaration, die auf die VB-Laufzeitbibliotheken zugreift, definiert werden. Zu beachten ist dabei, das die Deklaration der API-Funktion `VarPtrArray` an die entsprechend verwendete VB-Version angepasst werden muss.

```
'Deklaration für die Visual Basic 6 Version
Declare Function VarPtrArray Lib "msvbvm60.dll" Alias "VarPtr" ( _
      Ptr() As Any) As Long
```

Listing 19 API-Funktion VarPtrArray

Der Rückgabewert der vier Funktionen ist immer ein 32-Bit `Long`-Wert. Sollte der auf diese Weise ermittelte Pointer in einer `Long`-Variablen zwischengespeichert werden, ist bei dessen Verwendung im Aufruf einer API-Funktion unbedingt zu beachten, dass die Variable mit dem Schlüsselwort `ByVal` (detailierte Beschreibung von `ByVal` im Abschnitt „Praktische Umsetzung“) übergeben wird. Andernfalls wird nicht der gewünschte Wert, sondern der Pointer auf die Variable in der sich der Wert befindet übermittelt. [45, 48, 49]

CopyMemory

Der Zugriff auf den Wert einer Variablen, der an der Speicherstelle abgelegt ist, auf die ein Pointer verweist, ist auf direktem Wege unter VB nicht möglich. VB selbst stellt für Speicheroperationen, die mit vorhandenen Pointern durchgeführt werden, leider keine Funktion zur Verfügung. Glücklicherweise steht die API-Funktion `CopyMemory` zur Verfügung. Die Deklaration von `CopyMemory` lautet:

```
Declare Sub CopyMemory Lib "kernel32.dll" Alias "RtlMoveMemory" ( _
      pDst As Any, _
      pSrc As Any, _
      ByVal ByteLen As Long)
```

Listing 20 API-Funktion CopyMemory

`CopyMemory` kopiert, wie der Name schon sagt, einen Speicherblock in den eines anderen. Der Member `pDst` repräsentiert den Pointer auf den Zielspeicher und `pSrc` den Pointer auf den Quellspeicher. Des Weiteren enthält das Element `ByteLen` die Anzahl der zu kopierenden Bytes. Mit Hilfe dieser Funktion kann man den Speicherinhalt von einer Adresse in eine vorab deklarierte Variable des betreffenden Datentyps kopieren. [45, 46]

Handle

Ein Handle dient zur Identifikation und Verwaltung digitaler Objekte. Dabei verweist ein Handle auf eine interne Liste in der die verschiedensten Informationen und

Eigenschaften der Objekte gespeichert sind. Bedeutung gewinnt ein Handle, wenn es in einem bestimmten Kontext als Parameter an eine API-Funktion übergeben wird.

GetObjectAPI - Funktion

Wird in einen Speicherbereich eine Bitmap als Datei eingeladen, ist diese im Sinne von Windows ein Objekt und verfügt über ein Handle, das sogenannte Bitmaphandle. Durch Übergabe dieses Handles an die API-Funktion `GetObjectAPI`, können die Informationen und Eigenschaften des angegebenen Grafikobjekts ermittelt werden. Die Deklaration der Funktion lautet wie folgt:

```
Declare Function GetObjectAPI Lib "gdi32" Alias "GetObjectA" ( _
      ByVal hObject As Long, _
      ByVal nCount As Long, _
      lpObject As Any) As Long
```

Listing 21 API-Funktion GetObjectAPI

Das Element `hObject` erwartet das Handle eines Grafikobjektes, dessen Informationen ermittelt werden sollen. `lpObject` ist ein Zeiger auf einen Puffer, der nach Aufruf der Funktion, Informationen über das durch `hObject` bezeichnete Objekt zurückgibt und `nCount` die Größe des mit `lpObject` festgelegten Puffers in Byte. In diesem Fall ist das angegebene Grafikobjekt eine Bitmap, die Information in Form einer Struktur Namens BITMAP zurück gibt. Diese hat folgenden Aufbau:

```
Type BITMAP
   bmType As Long
   bmWidth As Long
   bmHeight As Long
   bmWidthBytes As Long
   bmPlanes As Integer
   bmBitsPixel As Integer
   bmBits As Long
End Type
```

Listing 22 Bitmap-Struktur

Der Parameter `bmType` beschreibt den Typ des Bitmaps, dessen Wert immer NULL sein muss. `bmWidth` beschreibt die Breite der Bitmap in Pixel und `bmHeight` die Höhe, ebenfalls in Pixel. `bmPlanes` enthält die Anzahl der Farbebenen und `bmBitsPixel` die Anzahl der benötigten Bits für jedes Pixel um den Farbwert des Pixels zu beschreiben. Das Element `bmWidthBytes` besitzt einen durch zwei teilbaren Wert der die Anzahl

der Bytes in jeder Bildzeile darstellt und `bmBits` enthält einen Pointer zu dem Speicherbereich in dem die Farbwerte des Bitmaps abgelegt sind.
Gerade die letzen beiden Parameter werden in Verbindung mit dem SaveArray bei der Suchfunktion von großem Nutzen sein. [46, 47, 50]

5.2.3. Praktische Umsetzung

Die Entwicklung der ActiveX-DLL wird mittels VB 6.0 umgesetzt. Da der Befehlssatz von VB 6.0 zur Realisierung der Suchfunktion nicht ausreicht, werden am Beginn des Skripts einige API-Funktionen deklariert, die später zur Umsetzung benötigt werden. Eine detaillierte Beschreibung dieser Deklarationen wurde bereits im Kapitel „Notwendige Funktionen“ vorgenommen.
Mit dem Befehl `Public Function Schnittpunkt` wird die Funktion gestartet. Der Ausdruck `Schnittpunkt` steht dabei für den Namen der Prozedur, gefolgt von der Argumentenliste, die in den Klammern steht und dem Rückgabetyp der Funktionsanweisung, der in diesem Fall vom Datentyp `Boolean` ist, also einen Wert True (Wahr) oder False (Falsch) zurückgibt.

```
Public Function Schnittpunkt( _
        ByVal File1 As String, _
        ByVal File2 As String, _
        ByRef x As Variant, _
        ByRef y As Variant, _
        ByVal deviate As Variant) As Boolean
```

Listing 23 Funktionsstart

Mit den Argumenten `File1` und `File2` wird der Speicherort der beiden Bitmap-Dateien (Screenshot der Testapplikation und das Referenzbild) übergeben. Die beiden Elemente `x` und `y` sind beim Aufruf der Funktion NULL und dienen später zur Übergabe der Schnittpunktkoordinaten. Dabei ist zu beachten, das die Variablen mit dem Schlüsselwort `ByRef` (By Reference) übergeben werden. Diese werden damit direkt referenziert. Es wird also kein Wert übergeben, sondern ein Pointer auf die Position der Variablen im Speicher. Das hat zur Folge, dass im Gegensatz zu `ByVal` alle Änderungen dieser Variablen in der bearbeitenden Funktion direkt auf die ursprüngliche Variable einwirken. Bei einer mit `ByVal` übergebenen Variable wird nur der Wert überreicht, das heißt es wird eine Kopie erstellt und nur diese an die Funktion weitergegeben. Eine Änderung innerhalb einer Funktion würde keine rückwirkenden

Auswirkungen auf die ursprünglich übergebene Variable besitzen. Mit dem letzten Element der Argumentenliste, das mit dem Namen `deviate` gekennzeichnet ist, wird die vom Benutzer zulässige Abweichung der Farbinformation vom Originalbild übergeben.

Die erste Aufgabe der Funktion ist das Laden der beiden Bitmaps. Mit dem VB-Befehl `LoadPicture` wird eine Bitmap als Datei in den Speicherbereich eingeladen und erhält ein gültiges Objekthandle. Durch die Übergabe dieses Handles an die API-Funktion `GetObjectAPI` werden nun die Informationen und Eigenschaften der Bitmap ermittel und in Form der BITMAP-Struktur zurück gegeben. Im nächsten Schritt muss ein ByteArray vorbereitet werden, dessen Deskriptor auf den in der BITMAP-Struktur referenzierten Speicherbereich verweist. Dazu ist der Arraydeskriptor selbst zu konstruieren. Benötigt wird ein zweidimensionales Array, wobei die `SAFEARRAY2D` Struktur zum Einsatz kommt:

```
With SA1
   .cDims = 2
   .fFeatures = 0
   .cbElements = 1
   .cLocks = 0
   .Bounds(0).lLbound = 0
   .Bounds(0).cElements = bmp1.bmHeight
   .Bounds(1).lLbound = 0
   .Bounds(1).cElements = bmp1.bmWidthBytes
   .pvData = bmp1.bmBits
End With
```

Listing 24 Struktur des zweidimensionalen Arrays

`.cDims` erhält den Wert zwei, da es sich um ein zweidimensionales Array handeln soll. Der Datentyp der Feldelemente ist Byte, der pro Element je eine Speicherzelle benötigt. Daher ist `.cbElements` auf eins zu setzten. Folglich erhält `.fFeatures` den Wert NULL, da Byte ein numerischer Datentyp ist. Sperrungen liegen bis jetzt nicht vor, wodurch `.cLocks` auch auf NULL gesetzt werden kann.

Da ein zweidimensionales Array konstruiert werden soll, sind für die Arraygrenzen auch zwei `Bounds` zu berücksichtigen. `.Bounds(0).lLbound` und `.Bounds(1).lLbound` müssen dabei für ein nullbasierendes Array auf NULL gesetzt werden. Da eine 24-Bit Bitmap vorliegt und dort jedes Pixel 3 Byte benötigt, ist folgende Bedingung für die Grenzen des Arrays zu beachten:

Bitmaphöhe [in Pixel] · (Bitmapbreite [in Pixel] · 3).

Die Höhe und die Breite der Bitmap sind schnell ermittelt, da diese in der BITMAP-Struktur vorliegen. Schaut man sich jedoch den Member `bmWidthBytes` dieser Struktur noch einmal genauer an, kann die Operation zur Berechnung der Breite wegfallen, da dieses Element die Anzahl der Bytes in jeder Bildzeile spezifiziert. Dadurch ergibt sich für die `Bounds` folgende Struktur:

```
.Bounds(0).lLbound = 0
.Bounds(0).cElements = bmp1.bmHeight

.Bounds(1).lLbound = 0
.Bounds(1).cElements = bmp1.bmWidthBytes
```

Listing 25 Bestimmung der Grenzen

Zum Schluss muss `.bmBits` noch `.pvData` zugewiesen werden. Zur Erinnerung, `.bmBits` referenziert den Speicherbereich, in dem die Farbwerte der Bitmap abgelegt sind und `.pvData` referenziert den Speicher in dem die Daten des Arrays gespeichert sind. Durch diese Aktion befindet sich später, bei der Zuweisung des Deskriptors, die Bitmap im ByteArray. Jetzt muss nur noch der neue Deskriptor wie folgt angebracht werden:

```
CopyMemory ByVal VarPtrArray(pic1), VarPtr(SA1), 4
```

Listing 26 Kopieren des Pointers

Mit der `CopyMemory` Funktion wird nun der Pointer der gerade definierten Struktur auf den ersten Pointer des Doppelpointer kopiert. Durch dieses Umkopieren des 4 Byte langen Pointers wird in Mikrosekunden eine beliebig große Bitmap in das ByteArray kopiert. Schneller kann diese Action, im Vergleich zu anderen Funktionen, nicht ausgeführt werden. Befindet sich die Grafik erstmal in einem Array, lassen sich Operationen mit dieser Methode an der Bitmap durchführen, die in wenigen Mikrosekunden abgearbeitet werden, wie beispielsweise die Abfrage der Farbinformation eines bestimmten Pixels.

Da bei der Suchfunktion zwei Bitmaps miteinander verglichen werden, werden auch zwei ByteArray benötigt. Die Schritte `LoadPicture`, `GetObjectAPI`, die Konstruktion des Arraydeskriptors und das kopieren des Pointers mit der `CopyMemory` Funktion müssen also für die zweite Bitmap wiederholt werden.

Interessant ist jetzt, wie die Bilddaten im Array gespeichert werden. Die folgende Abbildung soll das verdeutlichen:

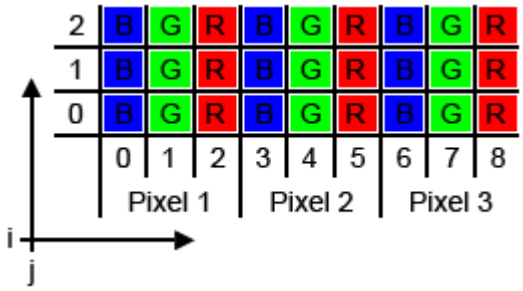

Abbildung 43 Anordnung der Bilddaten im Array

Wie man in der Abbildung deutlich erkennen kann, werden nicht nur die Bilddaten einer Bitmap von der linken unteren Ecke zur rechten oberen gespeichert. Die Anordnung der einzelnen Farbanteile ist ebenfalls vertauscht. Diese liegen nicht in der Reihenfolge RGB (Rot, Grün, Blau), sondern genau umgekehrt in einer BGR-Anordnung vor. Der Aufbau der Bilddaten im Array sollte deshalb stets präsent sein.
Möchte man nun beispielsweise den blauen Farbanteil eines Pixels, der in der unteren linken Ecke einer beliebigen Bitmap abgebildet ist abfragen, muss die Abfrage `pic(0,0)` heißen. Im Klartext bedeutet das, dass man im Array an der Position i = 0 und j = 0 den Byte-Wert abfragt, der den blauen Farbanteil des Pixels repräsentiert. Damit man die gesamte Farbinformation des linken unteren Pixels erhält müsste die Abfrage mit `RGB(pic(2,0), pic1(1,0), pic(0,0))` adressiert werden. Das zweite Pixel hat dann die Indizes `RGB(pic(5,0), pic1(4,0), pic(3,0))`. Allgemein könnte man also folgende Formulierung verfassen:

```
RGB(pic(i + 2, j), pic(i + 1, j), pic(i, j))
```

Listing 27 Allgemeine Formulierung zur Abfrage der Farbinformationen

Dabei steht für den roten Farbanteil `pic(i + 2, j)`, für den grünen `pic(1 + 1, j)` und für den blauen `pic(i, j)`.
Mit dieser komfortablen Ausgangsposition kann jetzt die Suchfunktion gestaltet werden, mit der es möglich ist, ein Referenzbild in einem Screenshot einer Testapplikation zu suchen (komplettes Skript: siehe Anhang). Dazu werden zwei `For`-Schleifen benötigt. Innerhalb der beiden Schleifen wird die Farbinformation (RGB-Wert) des linken unteren Pixels des Referenzbildes ermittelt. Danach müsste genau dieser Farbwert im Screenshot der Testapplikation gesucht werden, da es sich dabei um den linken unteren Rand des gesuchten Objekts handeln könnte. Gesucht wird im Bereich Höhe des Screenshots subtrahiert der Höhe des Referenzbildes und Breite des

Screenshots • 3 subtrahiert der Breite des Referenzbildes • 3. Nur in diesem Bereich kann das Objekt gefunden werden. Begonnen wird auch in der großen Grafik mit der Abfrage der Farbinformation des linken unteren Pixels.

Abbildung 44 Ausgangsposition

Die beiden Schleifen werden solange durchlaufen, bis die Farbinformationen der beiden Grafiken übereinstimmen. Dabei wird Zeile für Zeile der RGB-Wert eines jeden Pixels des Screenshots abgefragt und mit dem RGB-Wert des linken unteren Pixels des Referenzbildes verglichen.

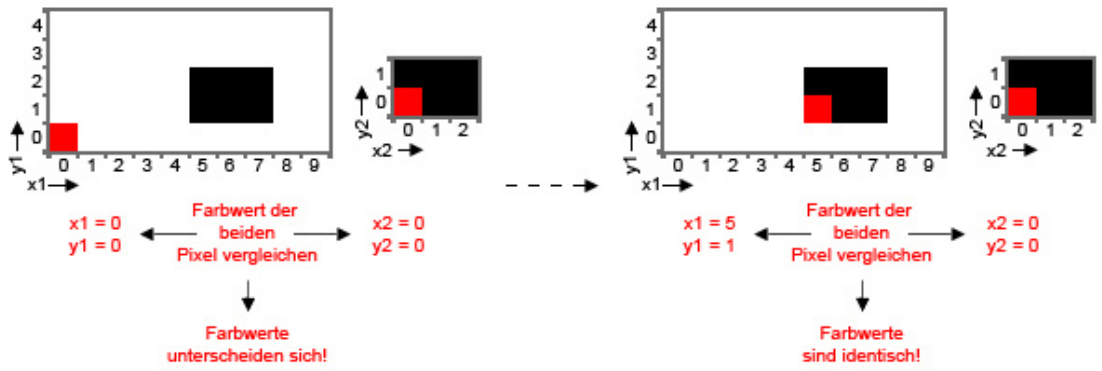

Abbildung 45 Pixelvergleich, bis Farbwerte identisch sind.

Stimmen die Farbinformationen überein, wäre es möglich, dass das gesuchte Objekt gefunden wurde. Ob das der Wirklichkeit entspricht muss die Prüfung, die durch zwei weitere `For`-Schleifen realisiert wird, beweisen. Der Bereich in dem Gesucht werden soll, ist durch die Höhe und Breite des Referenzbildes festgelegt. Jetzt werden allerdings nicht nur die Farbinformationen des folgenden Pixels im Screenshot abgefragt, sondern auch die Farbinformation des nächsten Pixels im Referenzbild. Bei erneuter Übereinstimmung der Farbwerte wird dieser Schritt wiederholt, bis das letzte Pixel vom Referenzbild (rechte obere Ecke) mit dem des Screenshots auf Gleichheit überprüft worden ist. Sollte sich vorher eine Abweichung der Farbwerte ergeben, wurden wahrscheinlich nur zufällige gleichfarbige Pixel gefunden. Demzufolge muss wieder die Farbinformation des ersten definierten Pixels im Referenzbild ermittelt

werden und die Suchfunktion am Folgepixel der ersten Übereinstimmung im Screenshot fortgesetzt werden.

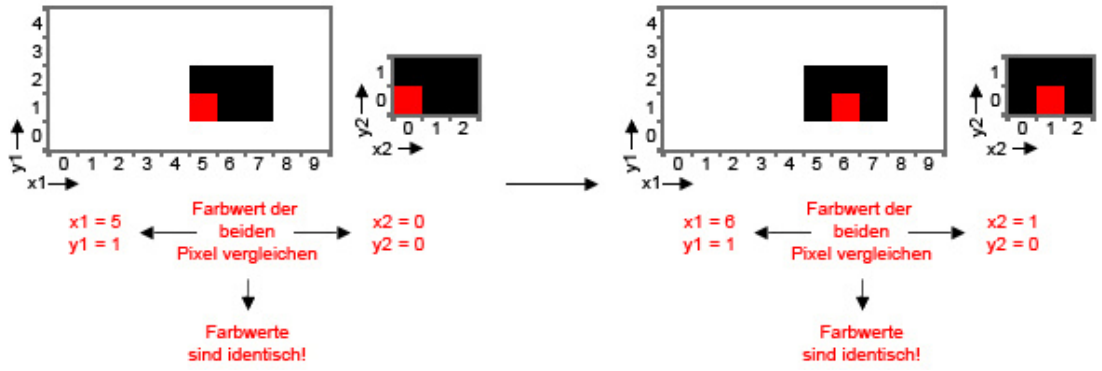

Abbildung 46 Prüfvorgang! Wurde das Referenzbild gefunden?

Bei einem durchgehend positiven Wertevergleich der Farbinformation, vom ersten bis zum letzten überprüften Pixel, war der Prüfvorgang erfolgreich und das gesuchte Objekt wurde gefunden.

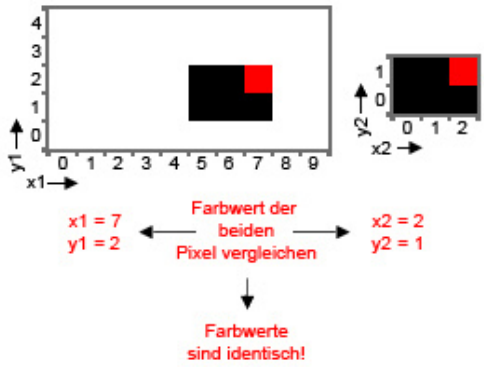

Abbildung 47 Prüfvorgang erfolgreich! Referenzbild wurde gefunden.

Der absolute Schnittpunkt, d.h. der Abstand des linken oberen Pixels des Referenzbildes, das sich innerhalb des Screenshots befindet, zum Pixel des linken oberen Randes des Screenshot, kann jetzt folgendermaßen berechnet werden:

```
abs_x = (w1 / 3) + 1
abs_y = bmp1.bmHeight - h1 - bmp2.bmHeight + 1
```

Listing 28 Berechnung des absoluten Schnittpunktes

`abs_x` steht dabei für die x-Koordinate und `abs_y` für die y-Koordinate des absoluten Schnittpunktes. Die Variable `w1` besitzt quasi den Wert in x-Richtung, an der das erste gemeinsame Pixel gefunden wurde. Hält man sich die Anordnung der Bilddaten im ByteArray noch einmal vor Augen, so stellt man fest, dass dieser Wert durch drei geteilt

werden muss, da im ByteArray ein Pixel in x-Richtung 3 Byte zur Darstellung des Farbwertes benötigt. Dieser Wert sollte anschließend mit eins addiert werden, da die Bilddaten im Array mit 0 beginnen und nicht wie in einem Grafikprogramm, in dem eine Bitmap-Datei zur Anzeige gebracht werden kann mit eins. Der Wert der Variablen `abs_y` errechnet sich aus der Bildhöhe des Screenshots subtrahiert der Höhe des Referenzbildes und der Variablen `h1`. `h1` besitzt den Wert des ersten gemeinsamen Pixels in y-Richtung und muss nicht durch drei geteilt werden. Der errechnete Wert von `abs_y` sollte aus den oben genannten Gründen ebenfalls mit eins addiert werden.
Da HP QTP später ein Klick-Event auf das Referenzobjekt ausführen möchte, wäre es besser nicht den absoluten Schnittpunkt zurückzugeben, sondern den Mittelpunkt des Referenzobjekts, bezogen auf den Abstand zum linken oberen Rand des Screenshots. Dazu muss lediglich der absolute Schnittpunkt mit der Höhe, bzw. Breite des Referenzobjekts, die jeweils durch zwei geteilt werden, addiert werden. `Round` rundet dabei die Zahl auf eine Ganzzahl ohne Dezimalstellen.

```
x = Round(abs_x + (bmp2.bmWidth / 2))
y = Round(abs_y + (bmp2.bmHeight / 2))
```

Listing 29 Runden der Schnittpunktvariablen

Zum Schluss muss nur noch die Variable `Schnittpunkt` auf `True` gesetzt werden und die Funktion mit dem Befehl `ExitFunction` beendet werden. Sollte der Funktionslauf ein negatives Ergebnis ermitteln, d.h. das Referenzobjekt befindet sich nicht im angegebenen Screenshot und kann dem entsprechend nicht gefunden werden, wird die Variable `Schnittpunkt` auf `False` gesetzt.
Da es in einigen Fällen zu kleineren Abweichungen, durch Unschärfe oder ähnliche Grafikfehler, bei der Darstellung einzelner Elemente innerhalb der Testapplikation kommen kann, sollte eine geringe Abweichung beim Vergleich der Farbwerte zulässig sein. Der Anwender muss dazu den maximal zulässigen Wert der Abweichung an die Suchfunktion übergeben. Die oben beschriebene Suchfunktion selber ändert sich dabei kaum. Lediglich die Abfrage der Farbinformation der einzelnen Pixel muss angepasst werden. Es soll nicht der gesamte RGB-Wert zweier Pixel verglichen werden, sondern die einzelnen Farbanteile dieser Pixel. Dadurch kann die Abweichung exakter überprüft werden.
Der Wert der Abweichung bezieht sich dabei auf die 256 Stufen, durch die ein Farbanteil angegeben wird. Beträgt die Abweichung beispielsweise den Wert 10 ist ein

Unterschied von maximal zehn Stufen in beide Richtungen zum originalen Farbanteil zulässig. Die folgende Abbildung soll das noch einmal verdeutlichen:

Abbildung 48 Zulässige Abweichung zweier Farbwerte

Zu beachten ist dabei, dass sich die Abweichung auf jeden der drei Farbanteile bezieht.

5.3. Aufrufmechanismus im Testautomatisierungswerkzeug

Bevor die Funktion zur Suche des Referenzbildes, das sich innerhalb des Screenshots einer Testapplikation befindet, in HP QTP verwendet werden kann, muss die ActiveX-DLL, in der die Funktion enthalten ist, im Betriebssystem, auf dem der Test durchgeführt werden soll, registriert werden.

Im Windows-Betriebssystem kann mittels der Datei REGSVR32.EXE eine ActiveX-DLL in die Registry des Betriebssystems eingetragen werden. Dabei wird in der Registry-Datenbank eine neue Class-ID angelegt, die auf die ActiveX-DLL-Datei verweist. Mit Hilfe dieser Class-ID können Programme wie HP QTP auf die ActiveX-DLL zugreifen und deren Funktion verwenden, bzw. ausführen.

Die Registrierung erfolgt über die Kommandozeile in der Eingabeaufforderung (cmd.exe), die unter Windows-Betriebssystemen mit Start → Ausführen… → Öffnen: cmd → OK zu starten ist. Mit dem folgenden Befehl wird danach die ActiveX-DLL registriert:

```
'Allgemein:
regsvr32 "Pfad\Dateiname.dll"

'Speziell:
regsvr32 "C:\SearchBitmap.dll"
```

Listing 30 Registrierung der ActiveX-DLL

Konnte die Registrierung erfolgreich durchgeführt werden, wird das Ergebnis durch eine entsprechend positive Mitteilung der Eingabeaufforderung zum Ausdruck gebracht. Zusätzlich kann die Registrierung mit dem kleinen Werkzeug Namens „VbsEdit“ überprüft werden. Hiermit wird die registrierte ActiveX-DLL aufgerufen und auf Wunsch kontrolliert.

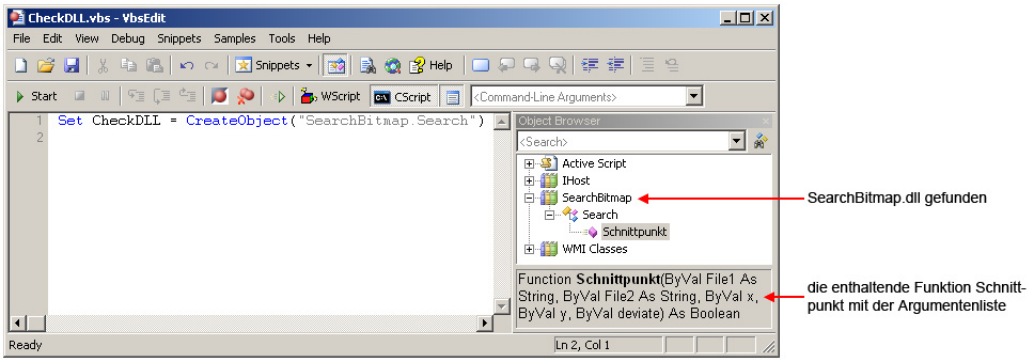

Abbildung 49 ActiveX-DLL Registrierung mittels VbsEdit überprüfen

Anschließend kann die Suchfunktion in HP QTP zum Einsatz kommen. Innerhalb eines Testscripts wird im „Expert View“ die ActiveX-DLL mit der entsprechenden Suchfunktion über die `CreateObject`-Funktion eingebunden. Genauer gesagt wird durch diesen Funktionsaufruf ein Verweis zur ActiveX-DLL erstellt. Dabei muss der genaue ActiveX-DLL-Name und die Klasse, in der die Funktion eingebettet ist, angegeben werden.

```
'Allgemein:
Set x = CreateObject("ActiveX-DLL-Name.Klasse")

'Speziell:
Set x = CreateObject("SearchBitmap.Search")
```

Listing 31 Verweis auf die ActiveX-DLL

Nachfolgend wird die Funktion `Schnittpunkt` (siehe Abschnitt „Praktische Umsetzung“) zur Referenzbildsuche aufgerufen und die Parameter werden übergeben. Zur Parameterliste gehört der Speicherort des Screenshots, in dem das Referenzbild gesucht werden soll. Darüber hinaus muss der Speicherort vom Referenzbild angegeben werden und die beiden Koordinaten (`x` und `y`) zur Lokalisierung des Referenzbildes. Die beiden Koordinaten sind bei der Parameterübergabe NULL und bekommen ihren Wert von der Suchfunktion (`Schnittpunkt`), der später zurück gegeben wird. Zum Schluss gehört in die Parameterliste die zulässige Abweichung, die das Referenzbild vom Original besitzen darf. Dem Parameter `deviate` (Abweichung) wird dabei ein Zahlenwert zugewiesen, der sich auf die 256 Stufen des RGB-Farbwertes bezieht (siehe Abschnitt: 5.1.3. Praktische Umsetzung → letzter Absatz).

```
sTemp = x.Schnittpunkt("C:\screenshot.bmp", "C:\referenz.bmp", _
                       x, y, deviate)
```

Listing 32 Funktionsaufruf und Parameterübergabe

Nach Beendigung der Suchfunktion wird der Wert True (Wahr) bei erfolgreicher Suche oder False (Falsch) zurückgegeben. Nur wenn ein Objekt im Screenshot der zu testenden Anwendung gefunden wurde, welches dem Referenzbild entspricht, wird der Wert der Variablen `x` und `y` geändert und kann in HP QTP abgefragt und beispielsweise für ein Klick-Event auf einen Button verwendet werden.

5.4. Gesamtbetrachtung

Zur Kontrolle der Funktion wird jetzt der Test aus dem Abschnitt „Fehlerhafte Objekterkennung“ wiederholt. In diesem Test sollte die Zoom-Funktion der Google-Map-Webseite überprüft werden, wobei die Objekterkennung von HP QTP zu Fehlern führte, bzw. das zu testende Objekt nicht erkannt werden konnte.
Im ersten Schritt muss das Referenzbild erzeugt werden. Dazu muss im Browser die Google-Map-Webseite geladen und auf dem Desktop für den Benutzer sichtbar sein. Jetzt kann mittels des CaptureIt-Tools das Referenzbild aus dem Screenshot der Google-Map-Webseite ausgeschnitten und im Dateisystem gespeichert werden.

Abbildung 50 Referenzbild für den Testlauf

Bevor mit der Aufzeichnung in HP QTP begonnen werden kann, sollte die ActiveX-DLL mit der Suchfunktion, also der alternativen Objekterkennung für HP QTP, im Windows-Betriebssystem registriert werden. Danach wird die zu testende Webseite in den gewünschten Ausgangszustand gebracht und HP QTP mit dem „Web“ Add-in gestartet. Anschließend wird ein neuer (leerer) Test geladen und mit einem Klick auf

den „Record“ Schalter die Aufzeichnung gestartet. Das Eingabefeld, der Ort der gesucht werden soll und der Klick auf den „Map-Suche“-Button in der Google-Map-Webseite wird wieder problemlos von HP QTP registriert und jeweils als ein Step im Testscript eingetragen. Nachfolgend wird wieder zum Testwerkzeug HP QTP gewechselt und der Test gestoppt. Die aufgezeichneten Testschritte können jetzt im „Keyword View“ oder im „Expert View“ überprüft und gegebenenfalls verändert oder angepasst werden. Im nächsten Schritt soll die Navigationleiste von Google-Map und dabei speziell der „Plus“-Button getestet werden. Dazu muss in HP QTP in den „Expert View“ gewechselt und das Skript angepasst werden.

Nach den beiden Steps für das Eingabefeld und das Klick-Event auf den Button, die im Testskript durch die Aufnahme bereits eingetragen sind, muss jetzt der Mechanismus für den Screenshot folgen. Bevor jedoch dieser Schritt eingesetzt werden kann, sollte der Webseite etwas Zeit gegeben werden, damit diese sich vollständig aufbauen kann und für den Screenshot komplett geladen ist. Mit dem Befehl `wait` und einer darauffolgenden Zeitangabe in Sekunden kann dieser Schritt im Testskript angegeben werden. Danach folgt die Aufforderung für das Auslösen des Screenshots mit der `CaptureBitmap`-Methode. Dabei wird ein Screenshot von der Webseite geschossen und im Dateisystem gespeichert. Bei der Vergabe des Dateinamens sollte kein spezieller Name angegeben werden. Ein allgemeiner Name wie beispielsweise „Webseite“ wäre geeigneter, da bei einem wiederholten Screenshot dieser File mit dem auf True gesetzten `OverrideExisting`-Attribut überschrieben werden kann. Das spart Speicherplatz im Dateisystem. Darüber hinaus muss die dazugehörige Dateierweiterung vom Typ Bitmap (.bmp) sein. Danach wird, wie bereits im Abschnitt „Aufrufmechanismus im Testautomatisierungswerkzeug“ beschrieben, ein Verweis zur ActiveX-DLL mit der `CreateObject`-Funktion erzeugt und die Funktion `Schnittpunkt` zur Referenzbildsuche aufgerufen. Dabei müssen die richtigen Parameter über den Speicherort des Screenshots und des Referenzbildes, die beiden Koordinaten `x` und `y` zur Lokalisierung des Referenzbildes und die zulässige Abweichung übergeben werden. Die Koordinaten sind bei der Skriptprogrammierung NULL und werden bei einem erfolgreichen Suchergebnis geändert. Darüber hinaus soll die zulässige Abweichung in diesem Test NULL sein. Eine Abweichung zum Original ist also nicht erlaubt. Da die Suchfunktion `Schnittpunkt` einen Boolean-Wert zurückgibt, kann für die beiden unterschiedlichen Ergebnisse ein einfacher Reporter für die Testauswertung erzeugt werden. Darüber hinaus kann ein Klick-Event auf den

„Plus“-Button nur durchgeführt werden, wenn der Rückgabewert True ist. Die beiden unterschiedlichen Reporter und die Ausführung des Klick-Events bei erfolgreicher Objektsuche sollten deshalb mit einer simplen Abfrage unterschieden werden.

```
' Verweis zur ActiveX-DLL
Set Bitmap = CreateObject("SearchBitmap.Search")
' Aufruf der Schnittpunktfunktion und Übergabe der Parameter
sTemp = Bitmap.Schnittpunkt("C:\screenshots\website.bmp", "C:\referenz\plusbutton.bmp", x, y, deviate)
' Ergebnisauswertung der Schnittpunktfunktion
If Not sTemp Then
   ' Erzeuge einen Reporteintrag mit „Warning“ (Warnung) → Objekt nicht gefunden
   Reporter.ReportEvent micWarning,"Searching Button", "Button not found!"
   ' Beende den „Run“
   ExitRun (1)
Else
   ' Erzeuge einen Reporteintrag mit „Passed“ (Erfolgreich) → Objekt gefunden
   Reporter.ReportEvent micPass, "Searching Button", "Button found! Coordinates: " & x & " / " & y
   ' Führe ein Klick-Event an den übergebenen Koordinaten durch!
   Window("Microsoft Internet Explorer").WinObject("Internet Explorer_Server").Click x, y
End If
```

Listing 33 Skriptcode zur alternativen Objekterkennung

Damit ist das Testskript komplett. Die Durchführung des Tests wird wieder durch den „Run“-Button im HP QTP Testwerkzeug gestartet. Dabei werden die einzelnen Schritte, der Reihenfolge nach, automatisch ausgeführt.

Zuerst wird im Eingabefeld der Ort eingegeben, der im Testlauf gesucht werden soll. Nachfolgend wird der „Map-Suche“-Button gedrückt und die Karte durch Google-Map neu geladen, wobei der gesuchte Ort zur Anzeige gebracht wird. Der nächste Schritt führt die CaptureBitmap-Methode aus, schießt damit einen Screenshot von der Webseite und speichert diesen im Dateisystem. Anschließend wird durch den Verweis auf die ActiveX-DLL, den Aufruf der `Schnittpunkt`-Funktion und der Übergabe der Parameter das Referenzobjekt im Screenshot gesucht. Nach wenigen Sekunden wird der Boolean-Wert der Funktion zurück gegeben und der Reporteintrag erzeugt. Da das Referenzbild gefunden wurde wird der Reporter „Passed“ (Erfolgreich) gewählt und der „Plus“-Button wird an den übergebenen Koordinaten durchgeführt. Die Abbildung 50 der Testergebnisse, die durch HP QTP nach Beendigung des Testlaufs erzeugt wurde, beweist, dass alle Schritte erfolgreich ausgeführt und das Referenzobjekt gefunden wurde. Demzufolge konnte der „Plus“-Button angeklickt und die Zoom-Funktion erfolgreich durchgeführt werden.

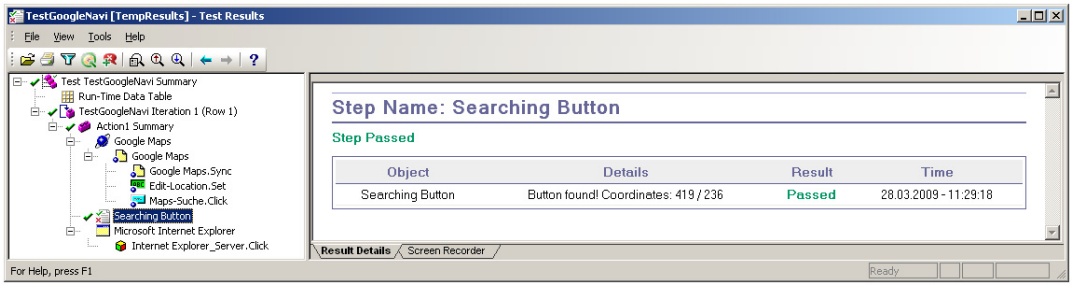

Abbildung 51 Testergebnis zur alternativen Objekterkennung

Damit konnte eine alternative Objekterkennung entwickelt und erfolgreich getestet werden, die immer dann eingesetzt werden kann, wenn HP QTP die Objekte in den Testapplikationen nicht erkennt. Dabei ist allerdings immer zu beachten, das die Referenzbilder im 24-Bit-Format abgespeichert werden. Deshalb sollte immer das CaptureIt-Tool verwendet werden, dass die Referenzbilder automatisch in diesem Format auf der Festplatte speichert. Dem zufolge kann die Farbtiefe bei den Testläufen vernachlässigt werden, da die Referenzobjekte durch das CaptureIt-Tool und die Screenshots der Testapplikationen durch die `CaptureBitmap`-Methode immer einen Farbwert von 24-Bit besitzen und damit eindeutig vergleichbar sind.

6. Zusammenfassung

In der heutigen Zeit nimmt der Zuwachs an Computersystemen im privaten, öffentlichen und wirtschaftlichen Bereich stätig zu. Die damit verbundenen Software-Systeme werden dabei immer umfangreicher, komplexer und anspruchsvoller. Unvorhersehbare und nicht geplante Wechselwirkungen werden jedoch auf Grund der zunehmend höheren Komplexität der Software-Systeme immer wahrscheinlicher. Dabei kommt es zu unterschiedlich schweren Schäden, die durch fehlerhafte Software Geld-, Zeit- oder Imageverlust im Unternehmen und der Wirtschaft verursachen und in einigen Fällen sogar zu Personenschäden führen können. Dementsprechend ist die Sicherung der Software-Qualität und das Testen der Software-Systeme ein wichtiger Bestandteil im Entwicklungsprozess, wobei die Herausforderung darin besteht, die Qualität der Software in einem angemessenen Umfang, in angemessener Zeit und zu angemessenen Kosten bei vertretbarem Risiko zu sichern. Der aufwändigste und umfangreichste Teil im Testprozess wird dabei vom Functional-Testing-Tools HP QTP unterstützt und automatisiert Funktions- und Regressionstest für alle wichtigen Softwareapplikationen und Umgebungen.

HP QTP nutzt das Konzept des automatisierten Testens mithilfe von Schlüsselwörtern und erleichtert das Erstellen und Verwalten von Tests, wobei die Testfälle mithilfe einer bestimmten Erfassungsmethode erstellt werden, bei der Abläufe und Objekte direkt aus den Applikationsbildschirmen erfasst werden. Bei einigen Anwendungen kann es jedoch zu Problemen mit der Objekterkennung durch HP QTP kommen. Wie im Abschnitt „fehlerhafte Objekterkennung“ beschrieben, gibt es keine Möglichkeit die unbekannten Objekte zu testen, bzw. führen die alternativen Möglichkeiten von HP QTP zu Fehlern und verfälschen die Testergebnisse enorm. Demzufolge wird der Umfang, in dem eine Softwareapplikation auf Fehler untersucht werden kann, stark reduziert.

Die entwickelte alternative Objekterkennung, die in dieser Arbeit vorgestellt wurde, stellt dabei eine gänzlich andere Herangehensweise zur Erkennung der Testobjekte bereit. Anhand eines vorgegebenen Vergleichsmusters kann ein bestimmtes Element überprüft und unabhängig von den Aussagen der HP QTP Objekterkennung bestimmt werden, ob und wo das Element tatsächlich in der Anwendung sichtbar ist. Das zusätzlich entwickelte CaptureIt-Tool ermöglicht dabei dem Automatisierer ein exaktes

zuschneiden des Referenzbildes, dass im automatisierten Funktions- und Regressionstest zur Anwendung kommt. Nach dem Auslösen des Screenshots der zu testenden Applikation wird im Testskript die Suchfunktion `Schnittpunkt` aufgerufen und das Referenzbild gesucht. Der Rückgabewert ist dabei immer ein Boolean-Wert, der die beiden unterschiedlichen Ergebnisse repräsentiert. Darüber hinaus werden die Schnittpunktkoordinaten des gesuchten Elements zurück gegeben, wenn es in der Testapplikation sichtbar ist. Demzufolge kann die alternativen Objekterkennung (Schnittpunktfunktion) eingesetzt werden, wenn HP QTP nicht in der Lage ist die Objekte in der Testapplikation zu erkennen.

Durch umfangreiche Testläufe mit Softwareapplikationen, die bei verschiedenen Unternehmen im Einsatz, bzw. im laufenden Betrieb sind, konnte die Zuverlässigkeit dieser alternativen Objekterkennung für HP QTP erwiesen werden. Die entwickelte Funktion ist dementsprechend Praxistauglich und kann in automatisierten Funktions- und Regressionstests, die mit HP QTP entwickelt werden, zum Einsatz kommen. Daher wird der Umfang, mit dem Softwareapplikationen getestet werden können, erweitert und die Qualität dieser Systeme wird gesteigert, wodurch das Risiko, das Fehler im operativen Betrieb auftreten, vermindert wird. Des Weiteren können Elemente getestet werden, die aufgrund der fehlenden Objekterkennung von HP QTP mühsam manuell getestet werden mussten. Dadurch werden Kosten und Zeit eingespart, die effektiver im Testprozess umgesetzt werden können.

A. Anhang

A.1. Skript: Tool zur Referenzbilderzeugung

```
Imports System.Runtime.InteropServices

Public Class Form1

   Private Const SRCCOPY As Integer = 13369376

   <DllImport("gdi32.dll")> _
   Private Shared Function BitBlt( _
      ByVal hDestDC As IntPtr, _
      ByVal x As Integer, _
      ByVal y As Integer, _
      ByVal nWidth As Integer, _
      ByVal nHeight As Integer, _
      ByVal hSrcDC As IntPtr, _
      ByVal xSrc As Integer, _
      ByVal ySrc As Integer, _
      ByVal dwRop As Integer) As Integer
   End Function

   <DllImport("user32.dll")> _
   Private Shared Function GetDC( _
      ByVal hwnd As Integer) As IntPtr
   End Function

   <DllImport("user32.dll")> _
   Private Shared Function ReleaseDC( _
      ByVal hwnd As Integer, _
      ByVal hdc As IntPtr) As Integer
   End Function

   'Variablen für Maus-Koordinaten
   Dim p1 As Point
   Dim p2 As Point
   'Variablen für Rechteckgröße (gestricheltes Rechteck)
   Dim DRx As Integer
   Dim DRy As Integer
   'Variablen für den Screenshot & zum Speichern des Referenzbildes
   Dim g1, g2 As Graphics
   Dim dc1, dc2 As IntPtr
   Dim img, ref As Image
   'Variablen für den Zoom
   Dim ZoomFaktor As Integer = 0
   Dim gs1, gs2 As Graphics
   Dim img2, img3 As Bitmap
   'Variable für Screenshot-Check
   Dim sCheck As Boolean = False
```

```
   Private Sub Form1_Load(ByVal sender As System.Object, _
      ByVal e As System.EventArgs) Handles MyBase.Load

      'Deaktivierung des Save-Buttons
      Save.Enabled = False
      'Deaktivierung der Zoom-Button
      ZoomIn.Enabled = False
      ZoomOut.Enabled = False
      'Deaktivierung des Clear-Buttons
      Clear.Enabled = False

   End Sub
```

```
   Private Sub Screenshot_Click(ByVal sender As System.Object, _
      ByVal e As System.EventArgs) Handles Screenshot.Click

      'Deaktivierung des Save-Buttons
      Save.Enabled = False
```

```
    'Deaktivierung der Zoom (-)
    ZoomOut.Enabled = False
    'ZoomFaktor auf 1 setzen
    ZoomFaktor = 1

    '*** Screenshot erzeugen

    'Form nicht sichtbar machen
    Me.Visible = False
    'Rechenzeit verschaffen, damit die Form beim Screenshot nicht zu sehen ist
    Call Wait(500)
    'Bitmap mit den Maßen des Desktop erzeugen
    img = New Bitmap(Screen.PrimaryScreen.WorkingArea.Width, _
                     Screen.PrimaryScreen.WorkingArea.Height, _
                     Imaging.PixelFormat.Format24bppRgb)
    g1 = Graphics.FromImage(img)
    'DC für den Screen erzeugen
    dc1 = GetDC(0)
    'DC für die Bitmap erzeugen
    dc2 = g1.GetHdc()
    'Bilddaten kopieren
    BitBlt(dc2, 0, 0, Screen.PrimaryScreen.WorkingArea.Width, _
              Screen.PrimaryScreen.WorkingArea.Height, dc1, 0, 0, SRCCOPY)
    'Screen-DC und Bitmap-DC freigeben
    ReleaseDC(0, dc1)
    g1.ReleaseHdc(dc1)
    'Zuweisen der Grafik an die PictureBox
    PictureBox1.Image = img
    'Form weider sichtbar machen
    Me.Visible = True

    'Clear-Button aktivieren
    Clear.Enabled = True
    'Zoom (+) aktivieren
    ZoomIn.Enabled = True

    'Screenshot-Check = True, da Screenshot gemacht und in der PictureBox sichtbar
    sCheck = True

End Sub
```

```
Private Sub Beenden_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Beenden.Click

    'Fenster schließen (Beenden)
    Me.Close()

End Sub
```

```
Private Sub PictureBox1_MouseDown(ByVal sender As Object, _
    ByVal e As MouseEventArgs) Handles PictureBox1.MouseDown

    'Refresh der PictureBox, damit der alte Markierungsrahmen gelöscht wird
    PictureBox1.Refresh()
    'Position, an der geklickt wurde, abspeichern
    p1 = e.Location
    p2 = e.Location

End Sub
```

```
Private Sub PictureBox1_MouseMove(ByVal sender As Object, _
    ByVal e As MouseEventArgs) Handles PictureBox1.MouseMove

    'Mausbewegung + linke Maustaste gedrückt, dann Anweisung ausführen
    If e.Button = Windows.Forms.MouseButtons.Left Then
        'Löschen des vorher gezeichneten Rechtecks
        ControlPaint.DrawReversibleFrame(New Rectangle(PictureBox1.PointToScreen(p1),_
            New Size(p2.X - p1.X, p2.Y - p1.Y)), Color.Black, FrameStyle.Dashed)
        'neue Koordinate bestimmen
        p2 = e.Location
        'zeichnen des neues Recktecks
        ControlPaint.DrawReversibleFrame(New Rectangle(PictureBox1.PointToScreen(p1),_
            New Size(p2.X - p1.X, p2.Y - p1.Y)), Color.Black, FrameStyle.Dashed)
```

```
    End If

End Sub
```

```
Private Sub PictureBox1_MouseUp(ByVal sender As Object, _
    ByVal e As MouseEventArgs) Handles PictureBox1.MouseUp

    'Ist der Screenshot in der PictureBox sichtbar?
    If sCheck = True Then

        'Save-Button aktivieren
        Save.Enabled = True

        'Ermitteln der Rechtecksgröße (Breite)
        DRx = (p2.X - p1.X) - 2
        'wenn DRx negativ, dann Anweisung ausführen
        If DRx < 0 Then
            DRx = (p1.X - p2.X) - 2
        End If

        'Ermitteln der Rechtecksgröße (Höhe)
        DRy = (p2.Y - p1.Y) - 2
        'wenn DRy negativ, dann Anweisung ausführen
        If DRy < 0 Then
            DRy = (p1.Y - p2.Y) - 2
        End If

    End If

End Sub
```

```
Private Sub Save_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Save.Click

    '*** Aktion Speichern (ref = Referenzbild)

    Dim ClipRectangle As Rectangle

    'Bilddaten kopieren (ACHTUNG! -> Zugrichtung des gestr. Rechtecks beachten)
    If (p1.X < p2.X) And (p1.Y < p2.Y) Then
        'Zugrichtung von oben links nach unten rechts
        ClipRectangle = New Rectangle(CInt(p1.X / ZoomFaktor) + 1, _
                                      CInt(p1.Y / ZoomFaktor) + 1, _
                                      CInt(DRx / ZoomFaktor), _
                                      CInt(DRy / ZoomFaktor))
    ElseIf (p1.X > p2.X) And (p1.Y > p2.Y) Then
        'Zugrichtung von unten rechts nach oben links
        ClipRectangle = New Rectangle(CInt(p2.X / ZoomFaktor) + 1, _
                                      CInt(p2.Y / ZoomFaktor) + 1, _
                                      CInt(DRx / ZoomFaktor), _
                                      CInt(DRy / ZoomFaktor))
    ElseIf (p1.X > p2.X) And (p1.Y < p2.Y) Then
        'Zugrichtung von oben rechts nach unten links
        ClipRectangle = New Rectangle(CInt((p1.X - DRx) / ZoomFaktor) - 1, _
                                      CInt(p1.Y / ZoomFaktor) + 1, _
                                      CInt(DRx / ZoomFaktor), _
                                      CInt(DRy / ZoomFaktor))
    Else
        'Zugrichtung von unten links nach oben rechts
        ClipRectangle = New Rectangle(CInt((p2.X - DRx) / ZoomFaktor) - 1, _
                                      CInt(p2.Y / ZoomFaktor) + 1, _
                                      CInt(DRx / ZoomFaktor), _
                                      CInt(DRy / ZoomFaktor))
    End If
    'Bitmap mit den Maßen des gestrichelten Rechtecks erzeugen
    ref = New Bitmap(ClipRectangle.Width, ClipRectangle.Height, img.PixelFormat)
    g2 = Graphics.FromImage(ref)
    g2.DrawImage(img, 0, 0, ClipRectangle, GraphicsUnit.Pixel)

    'SaveFileDialog
    Dim SaveFile As New SaveFileDialog
    'Parameter für den SaveFileDialog setzen
    With SaveFile
        .Title = "Save Bitmap"
        .FileName = ""
```

```
            .DefaultExt = "bmp"
            .Filter = "Bitmap (*.bmp)|*.bmp"
            .OverwritePrompt = True
        End With
        'OK-Button im SaveFileDialog gedrückt, dann Anweisung ausführen
        If SaveFile.ShowDialog() = Windows.Forms.DialogResult.OK Then
            ref.Save(SaveFile.FileName, System.Drawing.Imaging.ImageFormat.Bmp)
        End If

        'Refresh der PictureBox, damit der alte Markierungsrahmen gelöscht wird
        PictureBox1.Refresh()
        'Save-Button deaktivieren
        Save.Enabled = False

    End Sub
```

```
    Private Sub Wait(ByVal Milliseconds As Integer)

        'Variable "time" zur Zeitaufnahme.
        Dim time As Date
        'Addiert die angegebene Zahl vom Millisekunden zum aktuellen Zeit-Wert.
        time = Now.AddMilliseconds(Milliseconds)
        'Schleife wird durchlaufen bis "Now" erreicht ist.
        Do While time > Now
            Application.DoEvents()
        Loop

    End Sub
```

```
    Private Sub Clear_Click(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles Clear.Click

        'Clear -> PictureBox1 leeren
        PictureBox1.Image = Nothing
        'Clear-Button wieder deaktivieren
        Clear.Enabled = False
        'Screenshot-Check = False, da Screenshot in der PictureBox nicht mehr sichtbar
        sCheck = False
        'Save-Button deaktivieren
        Save.Enabled = False
        'Zoom-Button deaktivieren
        ZoomIn.Enabled = False
        ZoomOut.Enabled = False

    End Sub
```

```
    Private Sub ZoomIn_Click(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles ZoomIn.Click

        'Save-Button deaktivieren
        Save.Enabled = False

        'Eine Definition zum Zoom-Faktor befindet sich oben im Skrip!

        Select Case ZoomFaktor
            Case 1
                'Neues Bitmap mit den doppelten Maßen des Image1 erzeugen
                gs1 = PictureBox1.CreateGraphics()
                img2 = New Bitmap(img.Width * 2, img.Height * 2, _
                                  Imaging.PixelFormat.Format24bppRgb)
                gs1 = Graphics.FromImage(img2)
                'Image2 zeichnen
                gs1.DrawImage(img, New Rectangle(0, 0, img.Width * 2, img.Height * 2), _
                              New Rectangle(0, 0, img.Width, img.Height), _
                              GraphicsUnit.Pixel)
                'Image2 der PictureBox übergeben
                PictureBox1.Image = img2
                'Zoom-Faktor erhöhen
                ZoomFaktor = ZoomFaktor + 1

                'Zoom (-) aktivieren
                ZoomOut.Enabled = True
            Case 2
```

```
            'Neues Bitmap mit den doppelten Maßen des Image2 erzeugen
            gs2 = PictureBox1.CreateGraphics()
            img3 = New Bitmap(img.Width * 4, img.Height * 4, _
                              Imaging.PixelFormat.Format24bppRgb)
            gs2 = Graphics.FromImage(img3)
            'Image3 zeichnen
            gs2.DrawImage(img2, New Rectangle(0, 0, img2.Width * 2, img2.Height * 2), _
                          New Rectangle(0, 0, img2.Width, img2.Height),
                          GraphicsUnit.Pixel)
            'Image3 der PictureBox übergeben
            PictureBox1.Image = img3
            'Zoom-Faktor erhöhen
            ZoomFaktor = ZoomFaktor + 2

            'Zoom (+) deaktivieren
            ZoomIn.Enabled = False
        End Select

    End Sub
```

```
    Private Sub ZoomOut_Click(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles ZoomOut.Click

        'Save-Button deaktivieren
        Save.Enabled = False

        Select Case ZoomFaktor
            Case 2
                'Image1 der PictureBox übergeben
                PictureBox1.Image = img
                'Zoom-Faktor verringern
                ZoomFaktor = ZoomFaktor - 1

                'Zoom (-) deaktivieren
                ZoomOut.Enabled = False
            Case 4
                'Image2 der PictureBox übergeben
                PictureBox1.Image = img2
                'Zoom-Faktor verringern
                ZoomFaktor = ZoomFaktor - 2

                'Zoom (+) aktivieren
                ZoomIn.Enabled = True
        End Select

    End Sub

End Class
```

A.2. Skript: ActiveX-DLL zur Funktionserweiterung

```
Option Explicit

Private Type SAFEARRAYBOUND
   cElements As Long
   lLbound As Long
End Type

Private Type SAFEARRAY2D
   cDims As Integer
   fFeatures As Integer
   cbElements As Long
   cLocks As Long
   pvData As Long
   Bounds(0 To 1) As SAFEARRAYBOUND
End Type

Private Type BITMAP
   bmType As Long
   bmWidth As Long
   bmHeight As Long
   bmWidthBytes As Long
   bmPlanes As Integer
   bmBitsPixel As Integer
   bmBits As Long
End Type

Private Declare Function VarPtrArray Lib "msvbvm60.dll" Alias "VarPtr" ( _
   ByRef Ptr() As Any) As Long

Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" ( _
   pDst As Any, _
   pSrc As Any, _
   ByVal ByteLen As Long)

Private Declare Function GetObjectAPI Lib "gdi32" Alias "GetObjectA" ( _
   ByVal hObject As Long, _
   ByVal nCount As Long, _
   lpObject As Any) As Long
```

```
Private Function Schnittpunkt( _
   ByVal File1 As String, _
   ByVal File2 As String, _
   ByRef x As Variant, _
   ByRef y As Variant, _
   ByVal deviate As Variant) As Boolean

   Dim strFileName1 As String
   strFileName1 = File1

   Dim strFileName2 As String
   strFileName2 = File2

   Dim Bitmap1 As StdPicture
   Set Bitmap1 = LoadPicture(strFileName1)

   Dim Bitmap2 As StdPicture
   Set Bitmap2 = LoadPicture(strFileName2)

   Dim bmp1 As BITMAP
   Dim bmp2 As BITMAP

   Dim pic1() As Byte
   Dim pic2() As Byte
   Dim sa1 As SAFEARRAY2D
   Dim sa2 As SAFEARRAY2D

   GetObjectAPI Bitmap1, Len(bmp1), bmp1
   GetObjectAPI Bitmap2, Len(bmp2), bmp2
With sa1
      .cbElements = 1
      .cDims = 2
      .Bounds(0).lLbound = 0
```

```
   .Bounds(0).cElements = bmp1.bmHeight
   .Bounds(1).lLbound = 0
   .Bounds(1).cElements = bmp1.bmWidthBytes
   .pvData = bmp1.bmBits
End With

CopyMemory ByVal VarPtrArray(pic1), VarPtr(sa1), 4

With sa2
   .cbElements = 1
   .cDims = 2
   .Bounds(0).lLbound = 0
   .Bounds(0).cElements = bmp2.bmHeight
   .Bounds(1).lLbound = 0
   .Bounds(1).cElements = bmp2.bmWidthBytes
   .pvData = bmp2.bmBits
End With

CopyMemory ByVal VarPtrArray(pic2), VarPtr(sa2), 4

'Variablendeklaration
Dim a, b
Dim col1, col2
Dim deltaR, deltaG, deltaB
Dim locate_image As Boolean
locate_image = True

'Variablen für Bereichsdefinition
Dim h1, w1, h2, w2

'Variablen für Farbe
Dim i1, j1, i2, j2
i1 = 0
j1 = 0
i2 = 0
j2 = 0

'Variablen absoluter Schnittpunkt
Dim abs_x, abs_y

If deviate = 0 Then

   '*** Beginn Schleife 1 (deviate = 0)

   For h1 = 0 To bmp1.bmHeight - bmp2.bmHeight
      For w1 = 0 To (bmp1.bmWidth * 3) - (bmp2.bmWidth * 3)

         'Farbwert Pixel Image1
         col1 = RGB(pic1(i1 + 2, j1), pic1(i1 + 1, j1), pic1(i1, j1))
         'Farbwert Pixel Image2
         col2 = RGB(pic2(i2 + 2, j2), pic2(i2 + 1, j2), pic2(i2, j2))

         If col1 <> col2 Then
            w1 = w1 + 2
            i1 = i1 + 3
         Else

         '*** Beginn Schleife 2 (deviate = 0)

         For h2 = 0 To bmp2.bmHeight - 1
            For w2 = 0 To bmp2.bmWidth - 1

               'Farbwert Pixel Image1
               col1 = RGB(pic1(i1 + 2, j1), pic1(i1 + 1, j1), pic1(i1, j1))
               'Farbwert Pixel Image2
               col2 = RGB(pic2(i2 + 2, j2), pic2(i2 + 1, j2), pic2(i2, j2))

               If col1 = col2 And locate_image = True Then
                  locate_image = True
                  i1 = i1 + 3
                  i2 = i2 + 3
               Else
                  locate_image = False
                  i1 = i1 + 3
                  i2 = i2 + 3
               End If
```

```
            Next w2

            i1 = w1
            i2 = 0
            j1 = j1 + 1
            j2 = j2 + 1

         Next h2

         '*** Ende Schleife 2 (deviate = 0)

         '*** Beginn Auswertung (deviate = 0)

         If locate_image = True Then

            abs_x = (w1 / 3) + 1
            abs_y = bmp1.bmHeight - h1 - bmp2.bmHeight + 1

            'Wertzuweisung
            Schnittpunkt = True
            x = Round(abs_x + (bmp2.bmWidth / 2))
            y = Round(abs_y + (bmp2.bmHeight / 2))

            'Beenden
            Exit Function
         Else
            'Lokalisierungsindex wieder auf "True" setzen
            locate_image = True
         End If

         'Vergleichspixel Image2 wieder auf 0x0 setzen
         i2 = 0
         j2 = 0
         'Vergleichspixel Image1 setzen
         i1 = w1 + 3
         j1 = h1

         '*** Ende Auswertung (deviate = 0)

         w1 = w1 + 2

         End If

      Next w1

      i1 = 0
      j1 = j1 + 1

   Next h1

   '*** Ende Schleife 1 (deviate = 0)

Else

   '*** Beginn Schleife 1 (deviate <> 0)

   For h1 = 0 To bmp1.bmHeight - bmp2.bmHeight
      For w1 = 0 To (bmp1.bmWidth * 3) - (bmp2.bmWidth * 3)

         a = pic1(i1 + 2, j1)
         b = pic2(i2 + 2, j2)
         deltaR = Math.Abs(a - b)

         a = pic1(i1 + 1, j1)
         b = pic2(i2 + 1, j2)
         deltaG = Math.Abs(a - b)

         a = pic1(i1, j1)
         b = pic2(i2, j2)
         deltaB = Math.Abs(a - b)

         If deltaR > deviate And deltaG > deviate And deltaB > deviate Then
            w1 = w1 + 2
            i1 = i1 + 3
         Else
            '*** Beginn Schleife 2 (deviate <> 0)
```

```
            For h2 = 0 To bmp2.bmHeight - 1
               For w2 = 0 To bmp2.bmWidth - 1

                  a = pic1(i1 + 2, j1)
                  b = pic2(i2 + 2, j2)
                  deltaR = Math.Abs(a - b)

                  a = pic1(i1 + 1, j1)
                  b = pic2(i2 + 1, j2)
                  deltaG = Math.Abs(a - b)

                  a = pic1(i1, j1)
                  b = pic2(i2, j2)
                  deltaB = Math.Abs(a - b)

                  If deltaR <= deviate And deltaG <= deviate And _
                        deltaB <= deviate And locate_image = True Then
                    locate_image = True
                    i1 = i1 + 3
                    i2 = i2 + 3
                  Else
                    locate_image = False
                    i1 = i1 + 3
                    i2 = i2 + 3
                  End If

               Next w2

               i1 = w1
               i2 = 0
               j1 = j1 + 1
               j2 = j2 + 1

            Next h2

            '*** Ende Schleife 2 (deviate <> 0)

            '*** Beginn Auswertung (deviate <> 0)

            If locate_image = True Then

               abs_x = (w1 / 3) + 1
               abs_y = bmp1.bmHeight - h1 - bmp2.bmHeight + 1

               'Wertzuweisung
               Schnittpunkt = True
               x = Round(abs_x + (bmp2.bmWidth / 2))
               y = Round(abs_y + (bmp2.bmHeight / 2))

               'Beenden
               Exit Function
            Else
               'Lokalisierungsindex wieder auf "True" setzen
               locate_image = True
            End If

            'Vergleichspixel Image2 wieder auf 0x0 setzen
            i2 = 0
            j2 = 0
            'Vergleichspixel Image1 setzen
            i1 = w1 + 3
            j1 = h1

            '*** Ende Auswertung (deviate <> 0)

            w1 = w1 + 2

         End If

      Next w1

      i1 = 0
      j1 = j1 + 1

   Next h1

   '*** Ende Schleife 1 (deviate <> 0)
```

```
    End If

    CopyMemory ByVal VarPtrArray(pic1), 0&, 4
    CopyMemory ByVal VarPtrArray(pic2), 0&, 4

    Set Bitmap1 = Nothing
    Set Bitmap2 = Nothing

    'Rückgabe -> keinen Wert gefunden
    Schnittpunkt = False

End Function
```

Abbildungsverzeichnis

Listings

Abkürzungsverzeichnis

API	Application Programming Interface
BTO	Business Technology Optimization
DC	Device-Context
HP	Hewlett-Packard
IT	Informationstechnik
QC	Quality Center
QM	Qualitätsmanagement
QS	Qualitätssicherung
QTP	QuickTest Professional
VB	Visual Basic
VS	Visual Studio
DLL	Dynamic Link Library

Literaturverzeichnis

[1] Die Welt Online: Das bittersüße Lächeln des Bill Gates. URL: <http://www.welt.de/print-welt/article598661/Das_bittersuesse_Laecheln_des_Bill_Gates.html>, verfügbar am: 10.02.2009

[2] Wikipedia - Die freie Enzyklopädie: Toll Collect. URL: <http://de.wikipedia.org/wiki/Toll_Collect>, verfügbar am: 10.02.2009

[3] Wallmüller, Ernst: Software-Qualitätssicherung in der Praxis. - 1.Aufl. - München; Wien: Carl Hanser Verlag, 1990

[4] Wikipedia - Die freie Enzyklopädie: Qualitätsmanagementnorm. URL: <http://de.wikipedia.org/wiki/Qualitätsmanagementnorm>, verfügbar am: 12.02.2009

[5] profi.com AG: Software Qualitätssicherung. - 2009. - profi.com AG, Präsentation

[6] Wikipedia - Die freie Enzyklopädie: ISO/IEC 9126. URL: <http://de.wikipedia.org/wiki/ISO_9126>, verfügbar am: 12.02.2009

[7] Wohlrab, Lutz: Messungen zur Wartbarkeit von Software. - 2008. - Hochschule Mittweida (FH), Präsentation

[8] Prechelt, Lutz: Analytische Qualitätssicherung. - 2008. - Freie Universität Berlin, Institut für Informatik, Präsentation

[9] Gräbe, Hans-Gert: Software-Qualitätsmanagement. - 2005. - Universität Leipzig, Institut für Informatik, Präsentation

[10] Böhmer, Matthias: Statischer Test - Methoden und Möglichkeiten der statischen Analyse. - 2006. - FH Münster, Präsentation

[11] Wikipedia - Die freie Enzyklopädie: Statisches Software-Testverfahren. URL: <http://de.wikipedia.org/wiki/Statisches_Software-Testverfahren>, verfügbar am: 07.11.2008

[12] Wikipedia - Die freie Enzyklopädie: Dynamisches Software-Testverfahren. URL: <http://de.wikipedia.org/wiki/Testmethoden>, verfügbar am: 07.11.2008

[13] Wikiversity: Software-Test/Test-Techniken. URL: <http://de.wikiversity.org/wiki/Kurs:Software-Test/Test-Techniken>, verfügbar am: 07.11.2008

[14] Wikipedia - Die freie Enzyklopädie: Softwaretest. URL: <http://de.wikipedia.org/wiki/Softwaretest>, verfügbar am: 05.11.2008

[15] Wikipedia - Die freie Enzyklopädie: Black-Box-Test. URL: <http://de.wikipedia.org/wiki/Black-Box-Test>, verfügbar am: 10.11.2008

[16] Wikipedia - Die freie Enzyklopädie: White-Box-Test. URL: <http://de.wikipedia.org/wiki/White-Box-Test>, verfügbar am: 10.11.2008

[17] ISTQB: Grundlagen des Software-Testens. - 2008. - ISTQB, Präsentation

[18] Wikipedia - Die freie Enzyklopädie: Testautomatisierung. URL: <http://de.wikipedia.org/wiki/Testautomatisierung>, verfügbar am: 20.11.2008

[19] Seekamp, Jens: Automatisierter Software-Test unter Java. - 2007. - GEDOPLAN GmbH, Präsentation

[20] Computerwoche: Test-Werkzeuge für Software. URL: <http://wiki.computerwoche.de/doku.php/programmierung/software_testing_tools>, verfügbar am: 20.11.2008

[21] ITC Deutschland: <http://www.itc-germany.com/cms/upload/pdf/Vivit08/Vivit08_D%C3%BCsseldorf_QualityCenter_Blank.pdf>, verfügbar am: 03.12.2008

[22] Botros, Sinan: Projektübergreifendes Testmanagement am Beispiel des Produkts Quality Center in einem Finanzunternehmen. - 2006. - 74 S. Hannover, Universität, Fakultät für Elektrotechnik und Informatik, Bachelorarbeit, 2006

[23] von Hertell, Fabio: Automatisierte Software-Qualitätssicherung am Beispiel eines Frameworks für automatisiertes Testen. - 2009. - 56 S. Hamburg, Hochschule für Angewandte Wissenschaften, Fakultät Technik und Informatik, Bachelorarbeit, 2009

[24] Mercury - BTO News: Mercury Quality Center Extension für Visual Studio Team System. URL: <http://de.mercurybtonews.com/e_article000531281.cfm?x=b11,0,w>, verfügbar am: 25.11.2008

[25] harvard public relations: Mercury Pressemitteilung - Weltweite Optimierung von IT-Projekten. URL: <http://www.harvard.de/pressemeldungen/Mercury/2006/07-Mercury%20optimiert%20IT%20bei%20Marks%20and%20Spencer.pdf>, verfügbar am: 25.11.2008

[26] Hewlett-Packard: Business Technology Optimization (BTO) Software. URL: <https://h10078.www1.hp.com/cda/hpms/display/main/hpms_home.jsp?zn=bto&cp=1_4011_5__>, verfügbar am: 14.02.2009

[27] NOMADS: BTO-Team. URL: <http://wiki.informatik.hu-berlin.de/nomads/index.php/BTO-Team> verfügbar am: 25.11.2008

[28] contentmanager.de: Quality Center - Effiziente und fehlerminimierte Applikationsbereitstellung. URL: <http://www.contentmanager.de/magazin/news_h6055_mercury_quality_centereffiziente_und.html>, verfügbar am: 14.02.2009

[29] Hewlett-Packard: HP Quality Center. URL: <https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-127-24_4000_5__>, verfügbar am: 14.02.2009

[30] Hewlett-Packard: HP QuickTest Professional Software. URL: <https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-127-24^1352_4000_5__>, verfügbar am: 14.02.2009

[31] Scholz, Ralph: QuickTest Professional 9.0 - Schulungsunterlagen. - 1.Aufl. - Dresden: profi.com AG, 2006

[32] Oestereich, Michael: QuickTest Professional 9.0 - Schulungsunterlagen. - 1.Aufl. - Dresden: profi.com AG, 2006

[33] Hewlett-Packard Development Company: Using QuickTest Professional 9.2 - Instructor Guide. - USA: Hewlett-Packard Development Company, 2007

[34] Hewlett-Packard Development Company: Advanced QuickTest Professional 9.2 - Instructor Guide. - USA: Hewlett-Packard Development Company, 2007

[35] Hewlett-Packard Development Company: QuickTest Professional Help. - USA: Hewlett-Packard Development Company, 2007

[36] MSDN: Vorteile der Verwendung von DLLs. URL: <http://msdn.microsoft.com/de-de/library/dtba4t8b(VS.80).aspx>, verfügbar am: 05.01.2009

[37] MSDN: Exemplarische Vorgehensweise - Erstellen und Verwenden einer DLL (Dynamic Link Library). URL: <http://msdn.microsoft.com/de-de/library/ms235636(VS.80).aspx>, verfügbar am: 05.01.2009

[38] Wiki Unixboard.de: BMP. URL: <http://wiki.unixboard.de/index.php/BMP>, verfügbar am: 10.01.2009

[39] MSDN: Types of Bitmaps. URL: <http://msdn.microsoft.com/en-us/library/at62haz6.aspx>, verfügbar am: 10.01.2009

[40] MSDN: Bitmap Storage. URL: <http://msdn.microsoft.com/en-us/library/dd183391(VS.85).aspx>, verfügbar am: 10.01.2009

[41] Wikipedia - Die freie Enzyklopädie: Windows Bitmap. URL: <http://de.wikipedia.org/wiki/Windows_Bitmap>, verfügbar am: 10.01.2008

[42] Philipp, Teja: IMAGE REPRESENTATION. - 2004. - Referat im Proseminar "Computer Science Unplugged"

[43] dpunkt.verlag: Fotobearbeitung mit Paint Shop Pro X - Kapitel 2 - Das digitale Bild. URL: <http://www.dpunkt.de/leseproben/2310/Kapitel_2.pdf>, verfügbar am: 10.01.2009

[44] Wikipedia - Die freie Enzyklopädie: Programmierschnittstelle. URL: <http://de.wikipedia.org/wiki/Programmierschnittstelle>, verfügbar am: 20.01.2009

[45] MSDN: Windows API Reference. URL: <http://msdn.microsoft.com/de-de/library/aa383749(en-us).aspx>, verfügbar am: 20.01.2009

[46] ActiveVB: API-Katalog. URL: <http://www.activevb.de/rubriken/apikatalog/index-apikatalog.html>, verfügbar am: 20.01.2009

[47] ActiveVB: API-Wiki:Portal. URL: <http://activevb.de/cgi-bin/apiwiki/API-Wiki:Portal>, verfügbar am: 20.01.2009

[48] VB-fun.de: Einzelne Pixel setzen. URL: <http://www.vb-fun.de/cgi-bin/loadframe.pl?ID=vb/tutorial/tutorial003.shtml>, verfügbar am: 20.01.2009

[49] ActiveVB: Safearrays + Bitmaps. URL: <http://www.activevb.de/tutorials/tut_safearray/safearray.html#head>, verfügbar am: 20.01.2009

[50] vbarchiv: GetObject-Funktion. URL: <http://www.vbarchiv.net/api/api_getobject.html>, verfügbar am: 20.01.2009

[51] vbarchiv: BitBlt-Funktion. URL: <http://www.vbarchiv.net/api/details.php?id=bitblt>, verfügbar am: 28.01.2009

[52] Uni-Stuttgart: Blitting. URL: <http://w3studi.informatik.uni-stuttgart.de/~bischowg/DirectX/DX7DDrawBlitting.html>, verfügbar am: 28.01.2009

[53] VB-fun.de: GetDC. URL: <http://www.vb-fun.de/vb/api/GetDC.htm>, verfügbar am: 28.01.2009

[54] VB-fun.de: ReleaseDC. URL: <http://www.vb-fun.de/vb/api/ReleaseDC.htm>, verfügbar am: 28.01.2009

[55] VB-Seminar: Visual Basic Online Seminar - VB4 - Ereignisse. URL: <http://www.vb-seminar.de/vb_4.htm>, verfügbar am: 28.01.2009

[56] Doberenz, Walter; Gewinnus, Thomas: Visual Basic 2005 Kochbuch. 1.Aufl. - München: Carl Hanser Verlag, 2007

[57] Haupt, Horst F.: Visual Basic Referenz. - 2.Aufl. - Poing: Franzis-Verlag, 1999

[58] RRZN / Universität Hannover: Visual Basic 6.0 - Grundlagen Programmierung. 1.Aufl. - Nackenheim: HERDT-Verlag, 1999

Erklärung zur selbstständigen Anfertigung

Ich erkläre, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Mittweida, 24. April 2009

Philipp Wagner